VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky

BAKALÁŘSKÁ PRÁCE

2017 Ivo Jurišta

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra Informatiky

Online vizualizace provozních veličin automobilu
Online Visualization of Operating Variables for Cars and Truck

2017 Ivo Jurišta

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Ivo Jurišta**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Online vizualizace provozních veličin automobilu
Online Visualization of Operating Variables for Cars and Truck

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je vytvořit systém, který by zpracovával a online vizualizoval měřitelné veličiny pro osobní a nákladní vozidla. Cílem je shromažďovat a vizualizovat měřitelné údaje poskytované řídící jednotkou a to jak pro účely sledování okamžitého stavu, tak analýzy poruchovosti.

1. Seznamte se s problematikou měřitelných veličin u automobilů. Popište existující systémy, které danou problematiku řeší. Zpracujte rešerši těchto veličin a popište možnosti jejich získávání.
2. Proveďte analýzu požadavků cílových zákazníků, zejména na požadované výstupy kompatibilní s desktopovými i mobilními klienty.
3. Řešte napojení aplikace na měřící zařízení, které bude poskytovat získané veličiny pro vizualizaci a hodnocení. Implementujte databázi pro ukládání získaných dat. Použijte databázové systémy MySQL nebo Postgre SQL.
4. Implementujte vizualizaci naměřených dat, včetně vizualizace polohy vozidla v mapových podkladech. Implementaci realizujte jako webovou aplikaci s vhodně zvolenou technologií.
5. Výsledné řešení otestujte, srovnejte s konkurenčními aplikacemi, zhodnoťte dosažené výsledky a navrhněte možnosti dalšího rozšíření.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího bakalářské práce.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Radoslav Fasuga, Ph.D.**

Datum zadání: 01.09.2014

Datum odevzdání: 28.04.2017

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

VYSOKÁ ŠKOLA BÁŇSKÁ-TECHNICKÁ UNIVERZITA Fakulta elektrotechniky a informatiky 1D16 OSTRAVA

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta :

„Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.“

V Ostravě dne: 15.04.2017 Podpis:

Poděkování:

Rád bych poděkoval panu Ing. Radoslavu Fasugovi, Ph.D. za vedení, odbornou pomoc, připomínky a podmětné rady při zpracovávání bakalářské práce.

Ivo Jurišta

Abstrakt
Osazení sériově vyráběných automobilů elektronickými čidly řízenými centrální elektronickou jednotkou je dlouholetý standart. Po boku standartní produkce také vždy existoval menší proud alternativních projektů automobilů na zakázku, prototypů, domácích pokusů a studentských prací, či terénních, historických nebo jakýmkoliv způsobem nestandartních vozidel. Tato práce nachází inspiraci v modelu sériové výroby, ale zaměřuje se a hledá řešení právě pro menší alternativní projekty. Snaží se přijít s kompaktním řešením, které by bylo zjednodušenou, levnější a dostupnější verzí standartních elektronických modulů. Dále se zaměří na možnosti přenosu, analýzy a zobrazení dat takto získaných v navržené webové aplikaci pro mobilní a desktopová zařízení.

Klíčová slova: čtení elektronických dat z čidel, otevřená obousměrná komunikace, analýza v reálném čase, zobrazení v reálném čase, websockets, node.js, javascript, Bootswatch, FusionCharts, MySQL, Jade, Raspberry pi2

Abstract
The placement of an electronic sensors controlled with an electronic central unit into automobiles of the standard production has been common for many years. Also there have been existing always a smaller number of an alternative projects of a custom automobiles, a prototypes, a home DIY tryouts and student's works, or an offroad, historical or any type of a non-standard vehicles on the side of the standard production. This thesis gets an inspiration in the model of the standard production, but at the same time will search and focus on an alternative solution for the projects of a smaller scale. The thesis will try to bring a compact resolution, which shall be simplified, more affordable and better available version of the standard electronic modules. Furthermore, the thesis will focus on the possibilities of a transfer, an analysis and a vizualization of the data received, in the introduced web application for a mobile and desktop devices.

Keywords: a reading electronic data from sensors, an open bi-directional communication, a real time analysis, a real time vizualization, websockets, node.js, javascript, Bootswatch, FusionCharts, MySQL, Jade, Raspberry pi2

Seznam použitých zkratek a symbolů

AGPS – asistovaný globální polohovací systém
AJAX – asynchronní javascript a XML
API - rozhraní pro programování aplikací
ATS – čidlo teploty vcházejícího vzduchu
BARO – barometrické čidlo
CALM – pozemní bezdrátový typ komunikace
CAN-bus – komunikační sběrnice
CARB – rada životního prostředí (Kalifornie)
CCM – modul tempomatu
CEL – světlo kontroly motoru
CLS – čidlo hladiny chladící kapaliny
CNG – stlačený zemní plyn
CPC – kontroler mechanického pohonu
CSV – hodnoty čárkou oddělené
CTS – čidlo teploty chladící kapaliny
DB – databáze
DDC – diagnostický server Detroit
DDDL – název aplikace pro diagnostiku
DDEC – označení série spalovacích motorů
DIY – provést svépomocí
DTC – diagnostické kódy problémů
E85 – označení pohonné hmoty
ECM – elektronický kontrolní modul
ECU – electronická kontrolní jednotka
EEPROM – programovatelná pamět
EGO – čidlo obsahu kyslíku
EGR – redukce obsahu výfukových plynů
ETI – institut nářadí a výbavy
FPS – čidlo tlaku paliva
FS – systém souborů
FTS – čidlo teploty paliva
GPS – globální polohovací systém
GUI – grafické rozhraní uživatele
HP – výkonnost jednotky
HTML – typ značkovacího jazyka
HTTP – síťový protokol
IEEE – institut pro elektrotechnické a elektronické inženýrství
IS – informační systém
ISAM - metoda ukládání dat s možností rychlého vyhledávání
JSON – objektový zápis javascriptu

Seznam použitých zkratek a symbolů - pokračování

LIN – označení datové sběrnice
LPG – zemní plyn
LVDT – čidlo zatížení náprav
MCM – motorový kontrolní modul
MOST - označení datové sběrnice
MVC – model – vzhled – kontroler
MySQL – databázový systém
OBD – palubní diagnostika
OPS – čidlo tlaku oleje
OTS – čidlo teploty oleje
PHP – scriptovací jazyk
PID – identifikační kód, parametr ID
RPM – otáčky
SAE – společnost automobilových inženýrů
SRS – synchronní referenční čidlo
TBS – čidlo tlaku turbodmychadla
TCP – síťový protokol
TCS – modul kontrolující prokluz a smyk kol
TCU – kontroler převodovky
TFT - čidlo teploty převodové kapaliny
TPS - čidlo pozice pedálu akcelerace
TRS – časovací referenční čidlo
VLC – komunikace viditelného světla
VSS – čidlo rychlosti
WAWE – bezdrátový přístup automobilového prostředí
WS – síťový protokol
WSS - čidlo otáček kol
XML – rozšiřitelný značkovací jazyk

Seznam ilustrací

Seznam tabulek

OBSAH

Přílohy

1.ÚVOD

Automobilový průmysl využívá jevů z oboru elektrotechniky a informatiky již mnoho desítek let. V posledních letech však dochází nejen ke zvyšování požadavků v dané oblasti použití, ale i ke změnám mnoha faktorů otevírající nové možnosti. Digitalizace dat a zavedení internetu byly významným mezníkem v oblasti přenosu dat a od doby těchto změn je udržován relativně stabilní trend charakterizovaný několika podstatně se neměnícími rysy. Průběžně dochází ke zrychlování přenosu informace a to prakticky ve všech druzích mobilních sítí a navyšování kapacity množství přenesených dat. Také rapidním způsobem se daří expandovat a pokrývat těmito mobilními sítěmi nová území. Zároveň také dochází ke znatelnému zmenšování velikosti elektronických součástek a nárůstu výkonu procesorů, kapacity sběrnic a pamětí a to především díky neustálému objevování nových technologií. Pozitivní reakcí na tyto změny miniaturizace a integrace je produkce nových druhů mobilních zařízení, kontrolerů a čidel a jejich neustále klesající cena. Nové technologie v oblasti programovacích jazyků, komunikačních protokolů a softwarové sféry obecně jsou výsledkem rychlé adaptace na změny v hardwarových částí zařízení umožnující využití nových kapacit. Zatímco více jak deseti procentní meziroční vzrůst počtu elektronických modulů a čidel instalovaných v nových automobilech sériové produkce je důkazem rozmachu využití elektroniky v automobilovém průmyslu, souběžně se zde otevírají také prostory pro integraci elektronických součástek v oblasti středních a menších projektů.

1.1 Podmínky pro vývoj

Právě díky mnohem ekonomicky lepší dostupnosti elektronických součástek a modulů jsou produkovány nové typy mikroprocesorů a kontrolerů určené pro studijní potřeby, či amatérské projekty. Protože tyto projekty nabývají mnohdy nemalých rozměrů a fyzické části vysokých finančních hodnot, přirozeně se nabízí využití elektroniky, která je naopak finančně dostupnější, k jejich ochraně, monitorování i výzkumu. A protože již výše zmiňované mobilní technologie jsou dnes schopny informace pomocí elektronických čidel získané přenést v reálném čase prakticky v kterémkoliv místě, nastává ideální situace pro vývoj kompletních řešení přenosu dat v této oblasti.

1.2. Řešení a náplň

Bakalářská práce se zaměří na řešení pro projekty jako stavby elektromobilů, studentských formulí, přestavby a renovace historických vozidel a také testování nákladních a terénních automobilů. V první části práce, která je teoretického zaměření, se bude soustřeďovat na řešení a rozbor technologií běžné sériové výroby. V kapitolách dvě a tři představí stávající systémy, druhy čidel a jejich funkce a komunikace s okolím. Od počátku integrace elektronických modulů do automobilů v osmdesátých letech dochází k neustálému vývinu, obměnám a růstu a třetí kapitola ukončí popis těchto změn rozborem momentálního stavu a aktuálních technologií a možností. Vývojem samotného projektu se zabývají kapitoly čtyři až devět, které popíší průběh standartním vodopádovým modelem mající tyto části: specifikace zadání, analýza, návrh, implementace, testování a spuštění aplikace.

2. PROBLEMATIKA KOMUNIKACE

Nové osobní automobily vyšší třídy dnes patří k nejsložitějším strojům na naší planetě, obsahující sto milionů a více programového kódu. Tento údaj sám o sobě naznačuje složitost a obrovský rozměr oblasti digitální komunikace v automobilech. Popíši zde především základ, tedy veličiny a jejich hodnoty z vozidel získávatelné a monitorovatelné a později i jejich přenos do centrální jednotky vozidla a také vysílání v reálném čase mimo vozidla. Tuto konkrétní oblast komunikace sensorů umístěných ve vozidlech je možné rozdělit historicky, legislativně a technologicky.

2.1 Historie

V osmdesátých letech se začaly objevovat první, spíše signalizační, než řídící jednotky, které dokázaly upozornit na poruchu ve vozidle. V té době vozidla vyšší třídy dovedla automaticky informovat řidiče o stavu vybraných částí vozidel, které by jinak musel kontrolovat manuálně. Otevřené dveře, dálková světla, a další oznámení na palubní desce pomocí jednoduchých sensorů byl počátek standartního modelu tehdy jedné řídící jednotky komunikující se sadou čidel. On-board diagnostics (OBD), palubní diagnostika znamená schopnost vozidla provést svou vlastní analýzu stavu a reportovat problém. Tyto systémy technikům a majitelům vozidel umožňují přístup ke všem monitorovatelným subsystémům vozidla. Od počátku vývoje se množství dostupné informace ruzně liší. První verze OBD dokázaly jednoduše rozsvítit světlo indikující problém na palubní desce automobilu bez doložení jakýchkoliv dat popisujících problém. Zatímco moderní verze používají standartní rozhraní, sběrnici, dotazy a reporty, komunikují pomocí standartních kódu tzv. diagnostic trouble codes (DTC), ve standartních módech operací (tab. 1), uchovávají informaci v paměti, jakož také poskytují její realný přenos. Americká automobilka General Motors jako vůbec první v historii přisla na trh s termínem Engine Control Unit (ECU), tedy řídící jednotka motoru, který se později rozšířil po celém světě. Koncern Volkswagen již provozoval podobný systém na vozidlech se vstřikováním paliva. Chyby byly prezentovány technikům pomocí (CEL) Check Engine Lamp, tedy indikátor pro nutnost kontroly motoru, kterým řídící jednotka blikáním interpretovala konkrétní kód. V roce 1988 společnost automobilových inženýrů (SAE) doporučila výrobcům automobilů používat standartní konektor a standartní sadu kódu. Přesto za oficiálně první funkční standart se dá považovat teprve OBD II představený v roce 1991 a který začal být zákonem vyžadován až v roce 1996. Toto rozhraní bude sloužit pro detailní rozbor základního principu získávání hodnot veličin osobních i nákladních automobilů, jejich uchování a přenos.

Mode (hex)	Description
01	Show current data
02	Show freeze frame data
03	Show stored Diagnostic Trouble Codes
04	Clear Diagnostic Trouble Codes and stored values
05	Test results, oxygen sensor monitoring (non CAN only)
06	Test results, other component/system monitoring (Test results, oxygen sensor monitoring for CAN only)
07	Show pending Diagnostic Trouble Codes (detected during current or last driving cycle)
08	Control operation of on-board component/system
09	Request vehicle information
0A	Permanent Diagnostic Trouble Codes (DTCs) (Cleared DTCs)

tab. 1 Módy operací OBD II

2.2 Legislativa

Iniciativa skupin životního prostředí značně přispěla k rozvoji monitorování stavu a chování převážně motoru, palivové soustavy a spalovacího systému, tedy těm částem vozidel ovlivňující přímo životní prostředí v největším rozsahu. Takto došlo k první standartizaci povinných požadavků pro výrobce vozidel. Již zmíněné rozhraní OBD II vzniklo nepřímo jako reakce na rozhodnutí rady CARB, tedy Kalifornské rady pro životní prostředí v roce 1991, které odsouhlasilo nutnost standartu monitorování převážně množství spalin ve výfukových plynech. Následovala Evropská rada se stejným rozhodnutím v roce 2001 pro vozidla s benzinovými motory a v roce 2003 pro vozidla s naftovými motory. Korespondující sada čidel v této době, umístěna v prostoru motoru, palivové soustavě a výfukových potrubích monitorovala kvalitu spalování, pravidelný chod a popřípadě odchylky či chyby. Podstatnou součástí monitorovacího systému se stala sada testů, jejiž výsledky byl systém schopný zachytit a uchovat.

2.3 Technologie

Čtení dat, převážně hodnot daných fyzikálních veličin, ale i dalších informací o vozidle jako identita, rozměry, události či různé časové úseky mezi nimi je stále velice obsáhla oblast a ne celá podléhá legislativě a povinnému plnění korespondence se standartními rozhraními. Kromě povinného minima zde existují další skupiny sensorů a dat jimi produkovanými, které si konkrétní výrobce vozidla řeší individuálně a nemusí splňovat požadavky pro standartní komunikaci. Tyto informace obsahující nestandartní kódy jsou navíc veřejnosti nedostupné a hlavním bodem pro více výrobců automobilů je Equipment and tool institut (ETI), který udržuje dostupný seznam nestandartních kódů. Přesto ani tento institut nemá kompletní seznam všech existujících nestandartních kódů. Nastává zde situace kdy výrobci přichází s různými řešeními a navzdory snahy kontrolních orgánů k integraci hlavních výpočetních zdrojů v menší počet centralních kontrolerů se počet ECU v dnešních vozidel naopak zvyšuje. Průměrně se nachází až padesát oddělených kontrolních jednotek ECU v dnešním vozidle, v závislosti na typu a třídě. Na rozdíl od původního standartu OBD II kompletní systém dnešního vozidla monitoruje nejen hodnoty zakladních fyzikalních veličin, ale i vnitřní a vnější události ve vozidle, polohu, bezpečnost a chování řidiče, také začlenění vozidla v provozu a jízdní vlastnosti a styl.

2.4 Základní model

Bakalářská práce se zaměří na to jaké veličiny je možné a nejčastěji běžné ve vozidle monitorovat. Konkrétně popíše danou veličinu, její očekávané hodnoty za běžné situace a možné výchylky a anomálie. Čidlo které se ke čtení veličiny používá a napojení na řídící jednotku. Pro tento účel využije prvotního oficiálně uznaného standartního rozhraní OBD II a popíše zde získávání veličin a komunikaci s čidly právě přes toto rozhraní.[13]

Přenos po sběrnici

Po připojení diagnostického nářadí přes datový konektor zadá technik konkretní identifikační kód, tzv. parametr ID (PID). Tento parametr je na komunikační sběrnici CAN-bus (controller area network) rozpoznán korespondující jednotkou, která následně odpoví zpět na sběrnici s dotazovanou hodnotou. Hodnota je zobrazena na připojeném diagnostickém nářadí, kterým je i jednoduchý datový nahrávač (obr. 1), ruční čtečka s obrazovkou (obr. 2), nebo komplikované čtecí zařízení dodané výrobcem automobilu pro svůj vlastní produkt. Může mít i formu programu, instalovatelného do laptopu připojeného přes sériový port.

obr. 1 Datový nahravač

obr. 2 Ruční čtečka s obrazovkou

Velmi populární jsou kvalitní diagnostické aplikace a čtecí zařízení firmy Snap On (obr. 3).

obr. 3 Snap On nářadí

Datový kabel s koncovým šestnáctipinovým konektorem existuje ve verzi pro vozidla s 24 V instalací a pro vozidla s 12 V instalací. Sběrnice používá stejnojmenný komunikační protokol CAN pro sériovou komunikaci připojených zařízení, od jednoduchých vstup-vystupních jednotek až po výpočetní jednotky. Na sběrnici musí být připojena nejméně dvě komunikující zařízení. V období existence sběrnice CAN-bus vzniklo třináct norem upravujících chování a vlastnosti včetně hodnoty napájení. Protokol CAN zprostředkovávající přenos obsahuje čtyři základní vrstvy: fyzickou, transportní, objektovou a aplikační. Samotný přenos má základní jedenácti bitovou formu a rozšířenou dvacetibitovou formu přenášené informace. Typický dotaz (tab. 2) umístěný na sběrnici tzv.query se skládá z osmi bajtu, kde druhý obsahuje operační mód a třetí nese konkrétní PID, tedy oslovuje jedno unikátní zařízení, které zareaguje odpovědí se stejným PID a vyžádanou hodnotou. Další bajty zůstávají neobsazené v případě standartizované komunikace. [16]

	Byte							
PID Type	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
SAE Standard	Number of additional data bytes: 2	Mode 01 = show current data; 02 = freeze frame; etc.	PID code (e.g.: 05 = Engine coolant temperature)	not used (may be 55h)				
Vehicle specific	Number of additional data bytes: 3	Custom mode: (e.g.: 22 = enhanced data)	PID code (e.g.: 4980h)		not used (may be 00h or 55h)			

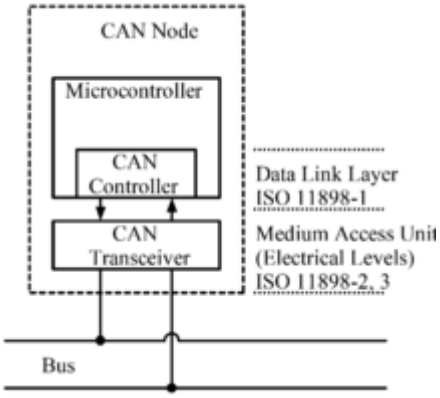
tab. 2 CAN bus Query

Stejného formátu je i daná odpověd, tzv. response (tab. 3). Vyžádána data jsou umísťována na pozici třetího až šestého bajtu v závislosti na délce dat.

	Byte							
PID Type	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
SAE Standard 7E8h, 7E9h, 7EAh, etc.	Number of additional data bytes: 3 to 6	Custom mode Same as query, except that 40h is added to the mode value. So: 41h = show current data; 42h = freeze frame; etc.	PID code (e.g.: 05 = Engine coolant temperature)	value of the specified parameter, byte 0	value, byte 1 (optional)	value, byte 2 (optional)	value, byte 3 (optional)	not used (may be 00h or 55h)
Vehicle specific 7E8h, or 8h + physical ID of module.	Number of additional data bytes: 4to 7	Custom mode: same as query, except that 40h is added to the mode value. (e.g.: 62h = response to mode 22h request)	PID code (e.g.: 4980h)		value of the specified parameter, byte 0	value, byte 1 (optional)	value, byte 2 (optional)	value, byte 3 (optional)
Vehicle specific 7E8h, or 8h + physical ID of module.	Number of additional data bytes: 3	7Fh this a general response usually indicating the module doesn't recognize the request.	Custom mode: (e.g.: 22h = enhanced diagnostic data by PID, 21h = enhanced data by offset)	31h	not used (may be 00h)			

tab. 3 CAN bus Response

Synchronizace je zajištěna kontrolerem, který na sběrnici prodlouží nebo zkrátí čas vymezený pro jeden nosný datový segment. Tato časová kontrola probíhá při každé náběžné hraně. Rychlost přenosu dat na sběrnici je možná až 1Mbit/s pro délku přenosu pod čtyřicet metrů.
Zařízení tzv.node připojené ke sběrnici (obr. 4) musí obsahovat mikroprocesor a kontroler, dále vysílač pro převod datového toku. Mikroprocesor získá přenášená data od kontroleru až v době kdy kontroler održí kompletní přenášenou zprávu a vyžádá si přerušení a následně přepošle data, která jsou mikroprocesorem zpracována. Mikroprocesor následně provede potřebná rozhodnutí.[17]



obr. 4 CAN Node

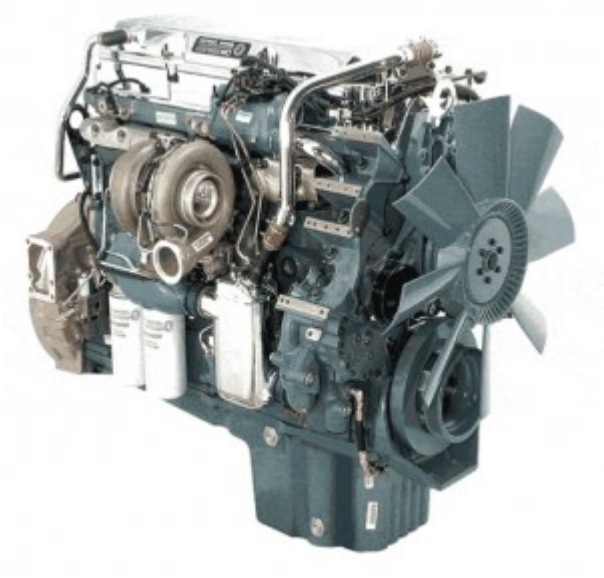
Motorická sekce

Skupiny čidel jsou rozmístěny v celém rozsahu vozidla a dělí se na dvě základní skupiny. Standartní, to je ta skupina čidel která snímá povinná data, to je data která jsou společná pro všechny výrobce automobilů a nestandartní, to je skupina čidel která přenáší data jednotlivými výrobci volitelná. První skupina čidel se zabývá přenosem veličin spojenými s pohonnou jednotkou, spalovací soustavou, palivovou soustavou a bezpečností. Zájem monitorování veličin se od samého počátku soustřeďoval právě na možnost analýzy spalin, kvality spalování a bezproblémového chodu motoru.

Nové typy pohonných jednotek, jako elektromotory, či hybridní soustavy uzpůsobené na spalování alternativních paliv typu compressed natural gas – stlačený zemní plyn (CNG), zemní plyn (LPG) a ethanol (E85) přinášejí potřebu monitorování nových veličin specifických pro konkrétní řešení. Klasický spalovací motor má dvě základní verze, zážehovou poháněnou benzinovými palivy a vznětovou poháněnou palivy naftovými. Obě verze pohonných jednotek mají část principu společný a liší se v podstatě zapálením pohonné směsi a strukturou vrchní části jednotky. Ve spodní části je umístěna hřídel, na této hřídeli ložiska obepnuta táhlem tzv.ojnicí na jejímž konci je přichycen píst. Tento píst souběžně při otáčení hřídele stlačuje a uvolňuje spalovací prostor, tedy prostor ve válci ve kterém je umístěn. Před celkovým stlačením dochází k nasávání paliva a vzduchu, resp.vstřikování paliva, v momentě kompletního stlačení k zapálení a při rozšiřování dochází k vypouštění paliva přeměněného ve spálený výfukový plyn. Tento proces je monitorován množstvím speciálních čidel snímajících teplotu, tlak, časování, počet otáček, obsah chemických látek, objem a elektrické napětí. Kromě jednoduchých čidel v této oblasti existují i sofistikované výpočetní jednotky, které provádějí výpočty v reálném čase a na sběrnici již hlásí pouze kód dané události či vypočtené hodnoty sledovaných veličin. Řídící jednotka motoru tyto informace zaznamenává a uchovává historii hodnot daných veličin. Tato jednotka také obsahuje nastavení, na kterém závisí výkon a spotřeba motoru a také emisní hodnoty chemických látek obsažených ve výfukových plynech, dusík(N2), vodní výpary(H2O), oxid uhličitý(CO2) mezi hlavními.

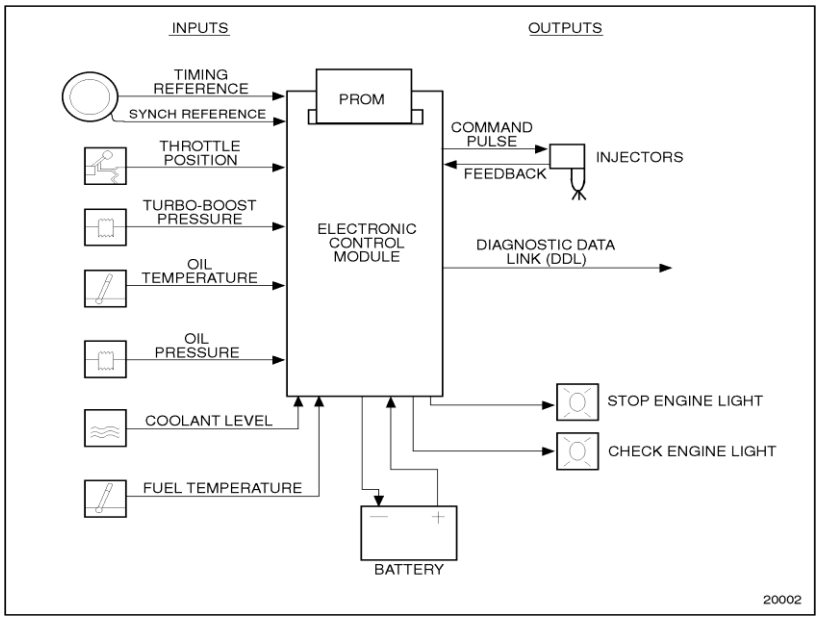
Detroit Diesel Series 60, 12.7L, 500HP

Pro lepší představu použijeme příklad (obr. 5) konkrétního motoru, označíme si umístění jednotlivých čidel a jejich konkrétní funkce. [3]



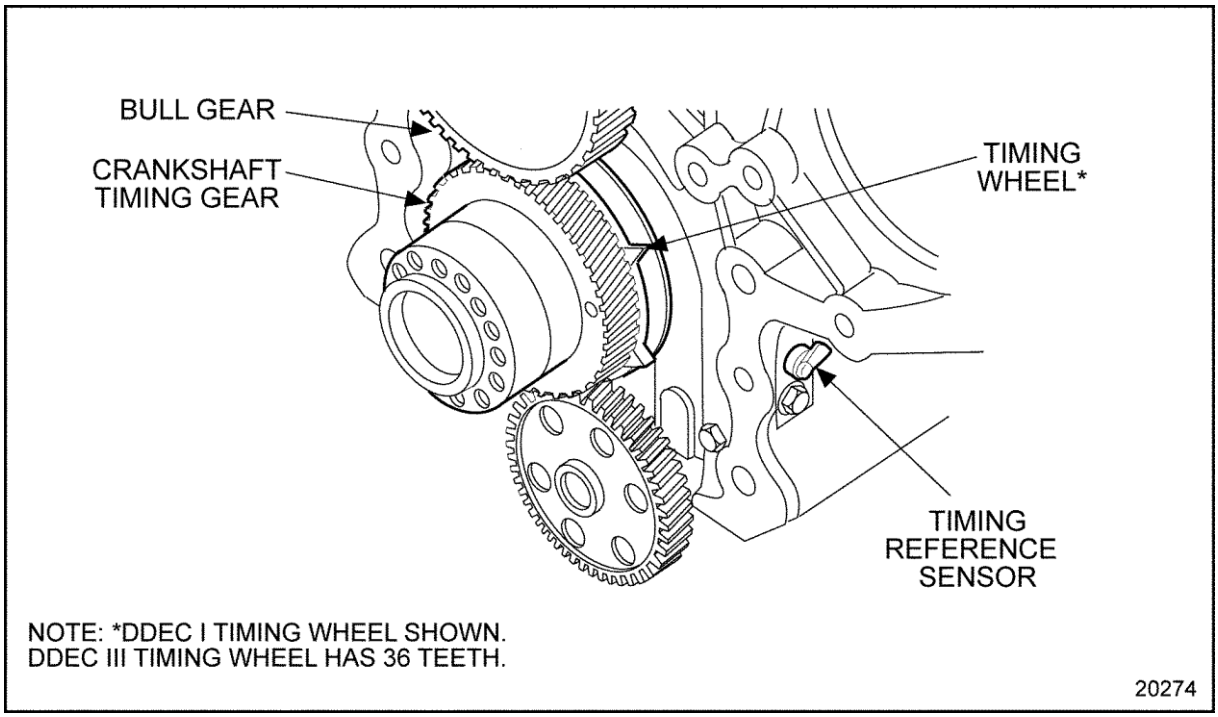
obr. 5 Motor Detroit Diesel

Výrobce Detroit Diesel při výrobě pohonné jednotky s označením serie 60 a rodiny DDECII jako první uvedl inovaci v podobě elektronické řídící jednotky za účelem zlepšení kvality výfukových plynů a dosažením lepší spotřeby paliva. Stává se tak průkopníkem v oblasti snímání hodnot veličin naftových pohonných jednotek a jejich digitalizace, uchování a přenosu. Stal se nejprodávanějším produktem své třídy po dobu třinácti let od roku 1987, inovace byla tedy velice úspěšná. Složitost a počet čidel motoru je ideální pro potřeby popisu (obr. 6) konkrétního příkladu na téma čidel a jejich funkcí na pohonných jednotkách.



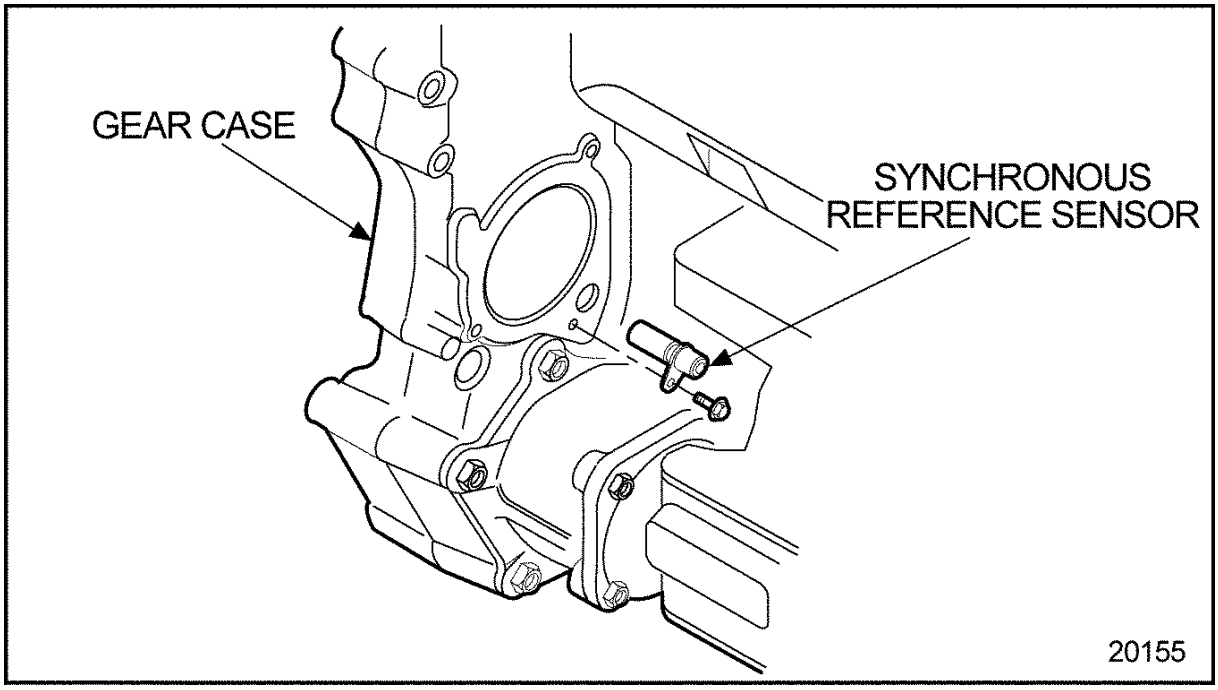
obr. 6 Schematický diagram DDEC II

Synchronous reference sensor (SRS) a timing reference sensor (TRS) je dvojice čidel pracujících společně a zajištujících korespondující chod vrchní (camshaft) a spodní (crankshaft) hřídele pohonné jednotky. Čidla indukčního typu, která jsou nahrazena u moderních motoru optickými čidly poskytují informaci o počtu otáček hřídelí. Časové referenční čidlo (obr. 7) je zodpovědné za získávání počtu otáček motoru, tedy údaj který je následně přenesen do elektronické řídící jednotky motoru a zobrazen na palubní přístrojové desce v kabině řidiče, jako číslo udávající počet otáček motoru za minutu (RPM) v otáčkoměru.



obr. 7 TRS čidlo

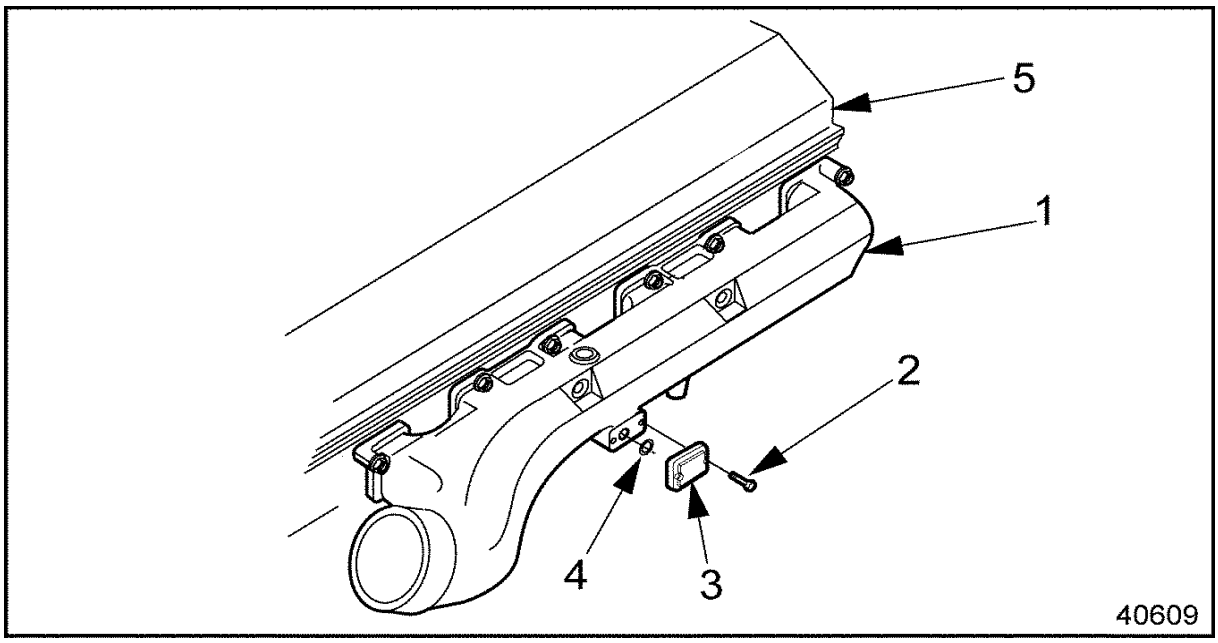
Anomálie která může nastat při poruše čidla je nepravidelné nebo žádné čtení otáček hřídele. Na základě informací získaných z tohoto čidla elektronická řídící jednotka vysílá signál dalším komponentům motoru jako vstřikovače paliva a dochází ke špatnému chodu pohonné jednotky. Tato situace nastáva v případě mechanického opotřebení čidla, které je vystavováno otřesům, vibracím a vysokým teplotám. Synchronní referenční čidlo (obr. 8) má identickou funkci a strukturu. Avšak pohonná jednotka je navržena takovým způsobem, že pro nastartování je potřeba obou čidel, pro následný provoz pouze čidla TRS. Čidla formulují informaci pomocí kódu 42 a 41, které zůstávají v paměti elektronické řídící jednotky i po vypnutí motoru.



obr. 8 SRS čidlo

Tyto kódy je možné také zobrazit na přístrojové desce. Před otočením klíče ve spínací skříni vozidla je nutno přidržet tlačítko pro vyžádání si kontrolních kódu ECM, které jsou následně signalizovány postupným blikáním jednoho signalizačního světla. Oranžová barva jsou obecné upozornění a červená vážné chyby.

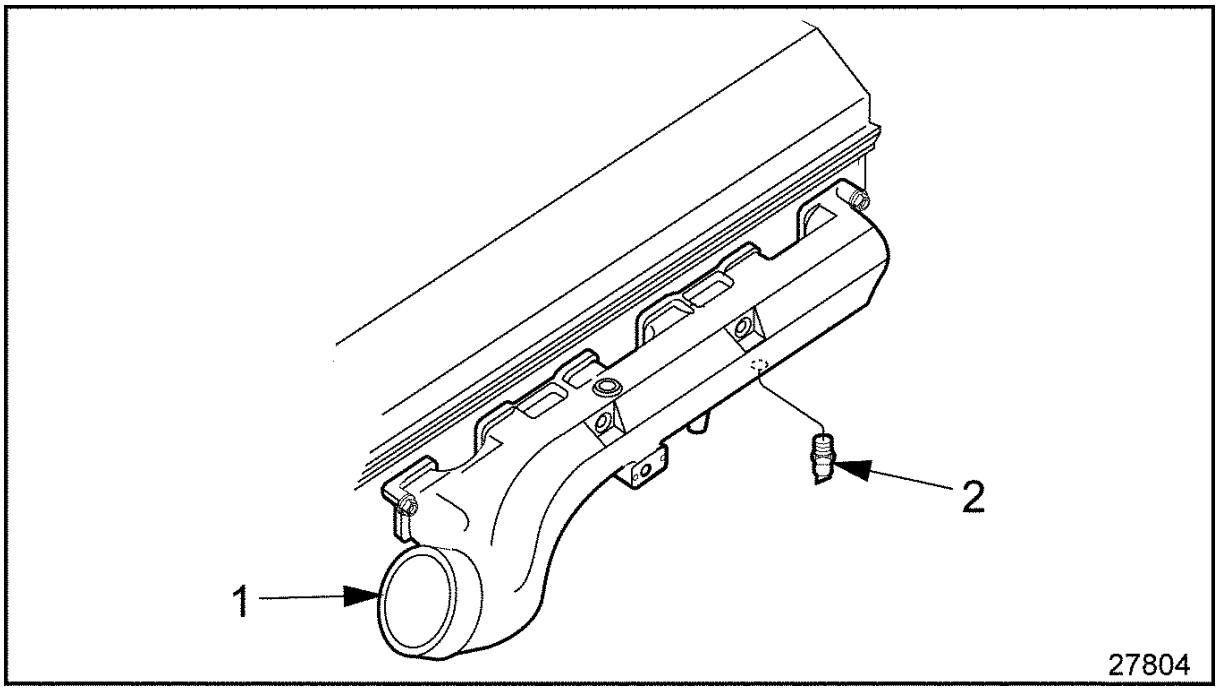
Turbo boost sensor (TBS) je tlakové čidlo umístěné ve vrchní části pohonné jednotky (obr. 9). Vysílá elektrický signál do ECM o objemu vzduchu vcházejícího dovnitř pohonné jednotky. ECM na základě této informace vypočítá množství paliva potřebného pro zážeh a reguluje tak hodnoty škodlivin ve výfukových plynech.



obr. 9 TBS čidlo

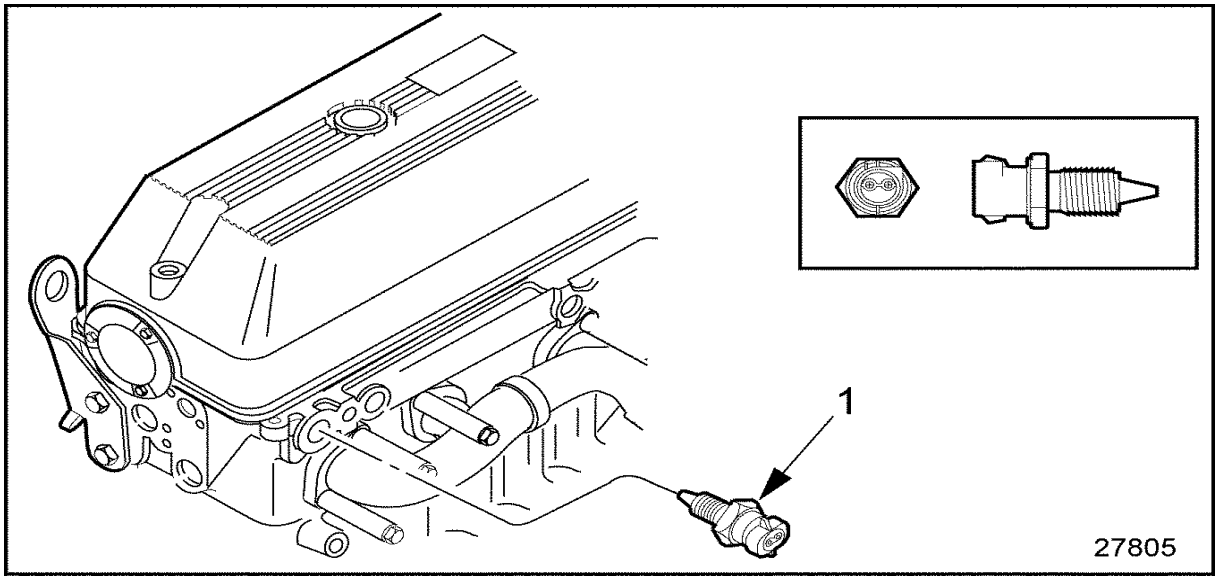
Anomálie které nastávají jsou zejména nízky tlak, způsobený netěsnostmi v okolí čidla a jako reakce nižší výkon motoru. Toto čidlo hraje významnou roli při tzv. tuningových operacích, kdy se majitele těchto pohonných jednotek snaží o dosažení většího výkonu motoru. Detroit diesel series 60 má standartní výkonnostní rozsah 420 až 515 HP, při tuningových operacích až 700 HP a více, což hraničí s rizikem exploze výfukových plynů, nebo destrukce turbodmychadla a následně celé pohonné jednotky.

Air inlet temperature sensor (ATS) je čidlo umístěno ve vrchní části motoru (obr. 10) a monitoruje teplotu vzduchu vcházejícího do sání pohonné jednotky. Vysílá elektrický signál do ECM na jehož základě elektronická řídící jednotka vytváří poměr pohonné směsi.



obr. 10 ATS čidlo

Tato informace napomáhá elektronické řídící jednotce zamezit tzv.bílému kouři, tedy špatně spálenému palivu v zimních měsících. V některých situacích společně s jinými událostmi dojde k sepnutí a rozepnutí relé pro ventilátor umístěný na chladiči chladící kapaliny v přední části pohonné jednotky.
Coolant temperature sensor (CTS) je čidlo umístěné na rohu vrchní části motoru (obr. 11) a monitoruje teplotu chladící kapaliny uvnitř pohonné jednotky.

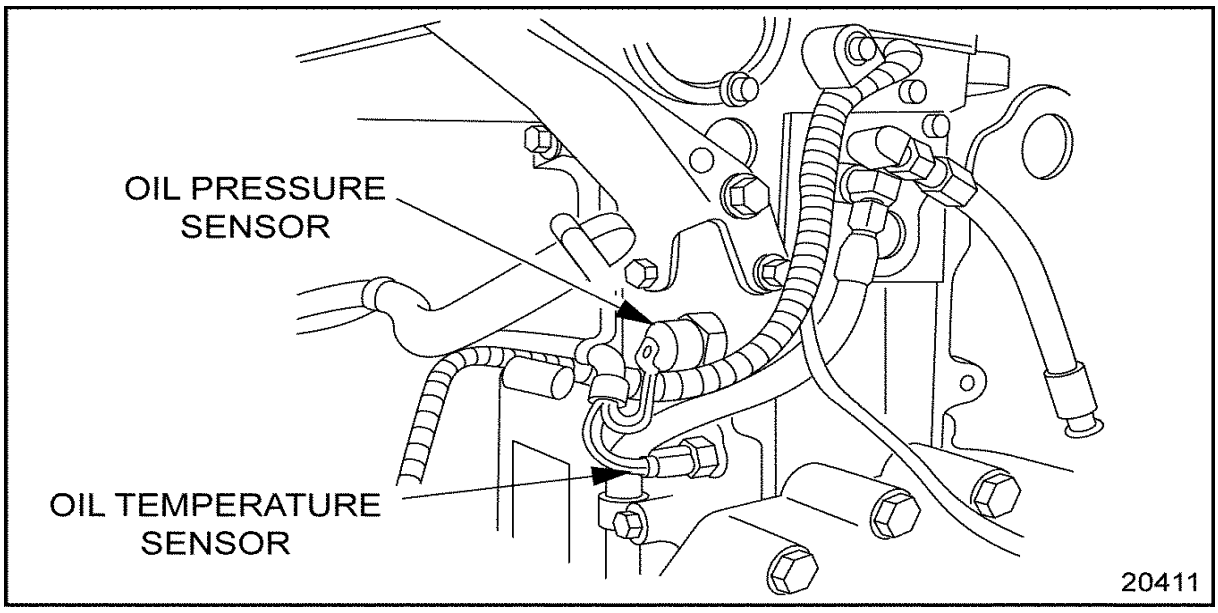


obr. 11 CTS čidlo

Informace získána pomocí tohoto čidla je přenesena do ECM a také využita k regulaci výkonu a množství škodlivin ve výfukových plynech. Zároveň je využita pro ochranu před přehřátím pohonné jednotky.

Ve stavu anomálie při překročení teplotních limitů ECM signalizuje na přístrojové desce oranžovým blikáním upozornění zvýšené teploty chladící kapaliny. Následuje koncové upozornění v podobě blikání varovného světla červenou barvou a v další fázi vypnutí motoru.

Coolant Level Sensor (CLS) je čidlo umístěné na nádobě chladící kapaliny v přední části vozidla. Modul vysílá elektrický signál do ECM pro indikaci hladiny chladící kapaliny. Nízká hladina chladící kapaliny aktivuje funkci zastavení motoru nebo varovnou funkci a není možno motor nastartovat.

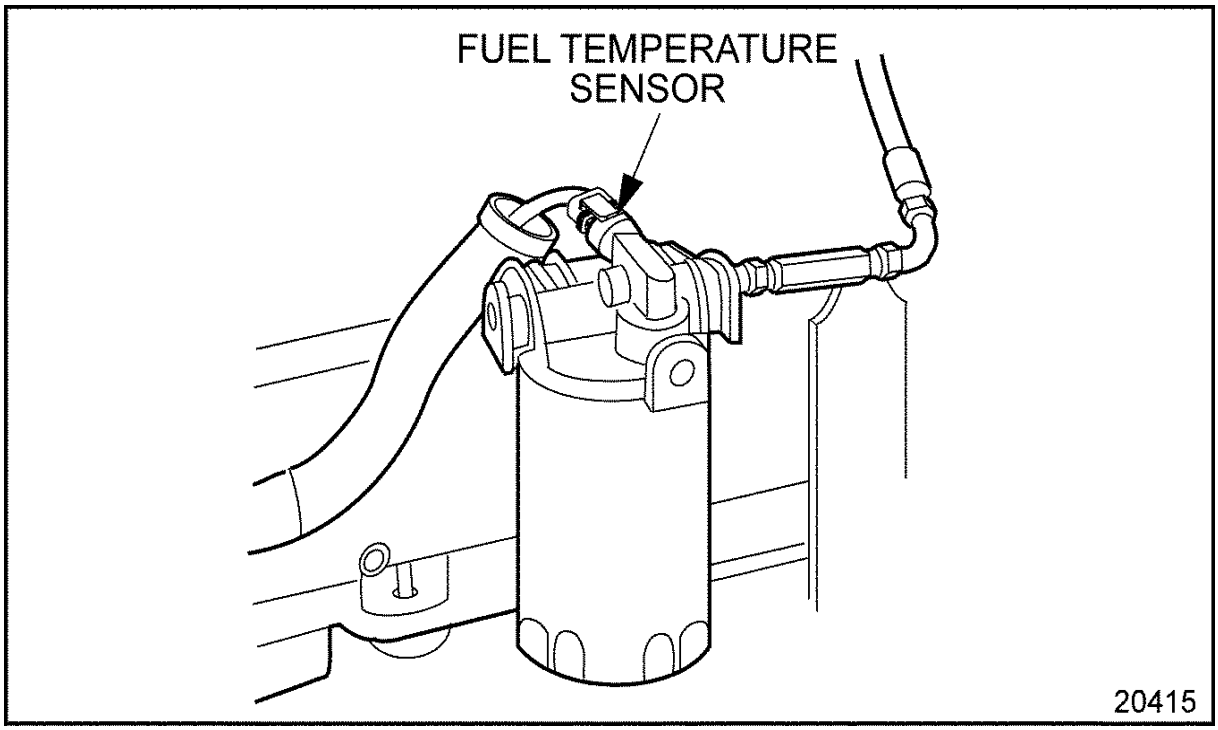


obr. 12 OPS a OTS čidla

Oil pressure sensor (OPS) je čidlo umístěné v zadní části motoru (obr. 12). Je to čidlo, které vysílá elektrický signál do ECM nesoucí hodnotu tlaku. Tato informace slouží k ochraně pohonné jednotky před poškozením nedostatečnou lubrikací interních částí systému. Anomálie opačného typu tedy překročení tlaku v olejové soustavě může způsobit poškození těsnících prvku v celé oblasti olejové soustavy pohonné jednotky. Hodnota tlaku je zobrazováná na přístrojové desce, kde doporučená hodnota výrobcem je 50 PSI. Tlak je regulován pomocí mechanického tlakového regulátoru.

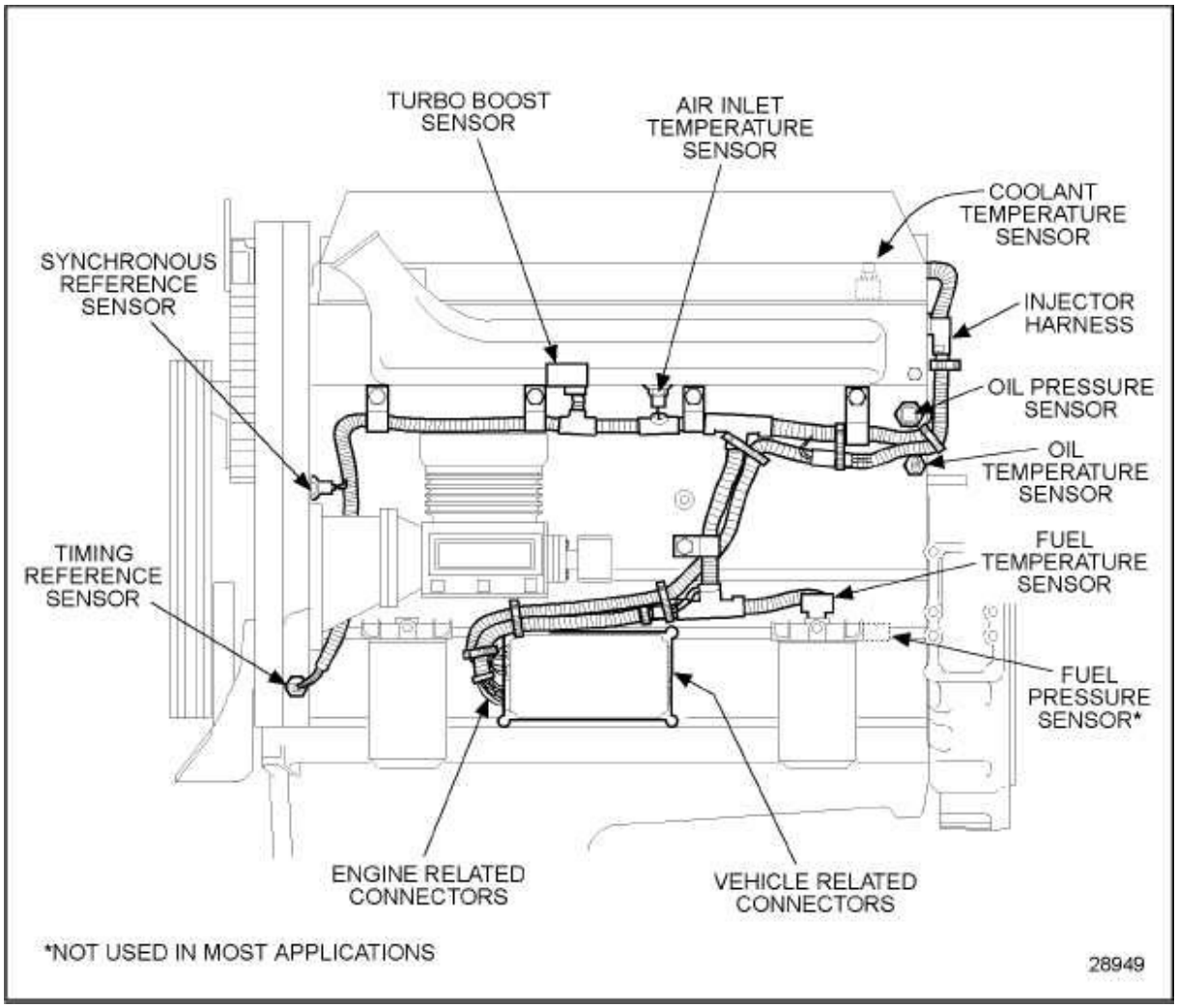
Oil Temperature Sensor (OTS) je čidlo pro snímání teploty oleje. Typickým místem je levý zadní roh bloku válců (obr. 12). OTS vysílá elektrický signál do ECM udávající teplotu oleje. Tato informace je využita pro regulaci otáček motoru pro lepší starty při chladném počasí a rychlejší zahřátí. Přesáhne-li teplota povolenou mez po dobu dvou sekund dochází k varování nebo zastavení motoru v závislosti na dalších hodnotách snímaných veličin.

Fuel temperature sensor (FTS) je čidlo umístěné na sekundárním palivovém filtru (obr. 13). Toto čidlo vysílá elektrický signál do ECM udávající teplotu paliva vstupujícího do palivového filtru.



obr. 13 FTS čidlo

Fuel pressure sensor (FPS) je čidlo umístěné na sekundárním palivovém filtru (obr. 14). Toto čidlo vysílá elektrický signál do ECM udávající tlak paliva vstupujícího do palivového filtru a není standartně dodáváno výrobcem pro všechny aplikace.



obr. 14 Přehled umístění čidel

Mezi další čidla, která nebývají součástí základní sady dodávané výrobcem ke každé pohonné jednotce patří například (BARO sensor) a (EGO) Exhaust Gas Oxygen sensor.

Baro Sensor je tlakové čidlo snímající externí barometrický tlak a vysílá elektrický signál do ECM obsahující informaci na základě které elektronická řídící jednotka vytváří potřebnou palivovou směs a reguluje časování zážehu palivové směsi ve spalovacím prostoru. S uživatelského hlediska to znamená, že se pohonná jednotka automaticky přizpusobuje nadmořské výšce ve které se automobil pohybuje.

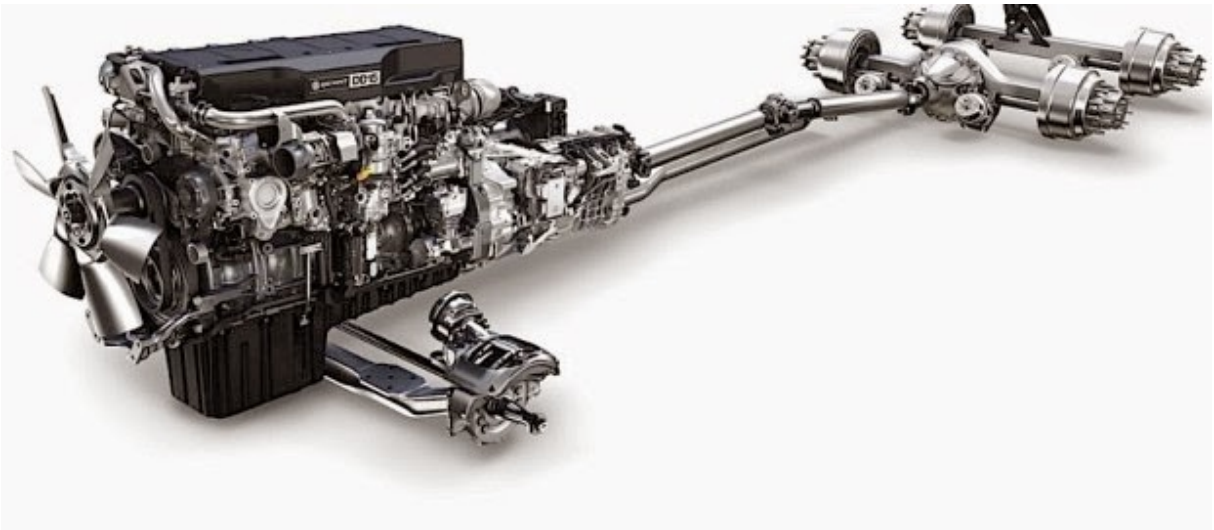
Exhaust Gas Oxygen sensor (EGO) je čidlo pro snímání hodnot kyslíku ve výfukových plynech a tato informace přenesena do ECM ve formě elektronického signálu udává, zda-li je palivová směs spálena dokonale a na základě této informace ECM vytváří poměr nasátého vzduchu a paliva pro zapálení v dalším cyklu. Má-li pohonná jednotka tzv.bohaté spalování, výfukové plyny obsahují méně kyslíku a naopak.[5]

ECM a komunikace s aplikací DDDL

Electronic control unit (ECM) tedy elektronická řídící jednotka (obr. 15) obsahuje 32-bitový mikroprocesor s vyjímatelnou pamětí EEPROM s možností uchování relativně malého množství dat, dále možností vymazání a přeprogramování. Po dobu existence pohonných jednotek rodiny DDEC bylo vyprodukováno celkem šest verzí ECM, v roce 1987 začínajících označením DDEC I a končící označením DDEC VI. Poslední verze ECM obsahuje dva kontrolery.

obr. 15 ECM Detroit DDEC

Motor control modul (MCM) tedy řídící jednotka motoru, má na starosti veškeré funkce předešlých typů ECM týkajících se čidel pohonné jednotky a komunikaci s nimi. Přibyla zde další řídící jednotka Common powertrain controller (CPC) tedy řídící jednotka komunikující s čidly v oblasti mechanického pohonu náprav (obr. 16), tzn. částí začínající bezprostředně za pohonnou jednotkou a obsahující převodovku, kardanové ústrojí včetně kloubů a diferenciálních skříní.

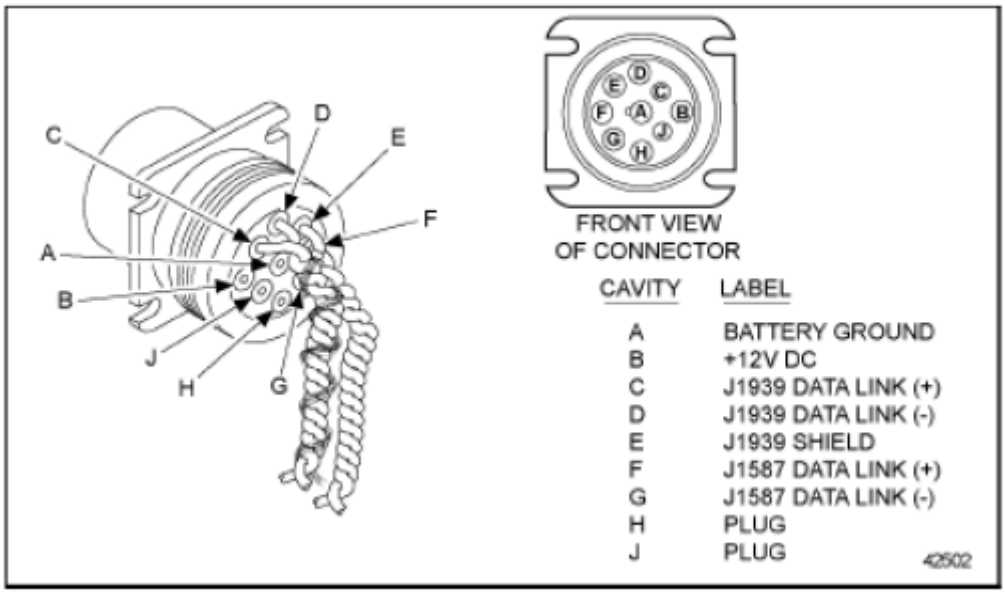


obr. 16 Detroit powertrain

Samotný fakt, že nejnovější typ ECM DDEC VI zdvojnásobil počet modulů potvrzuje na začátku bakalářské práce zmíněný trend rostoucího počtu ECM v kontrolních systémech dnešních vozidel. Níže se bakalářská práce zabývá také stručným popisem multi ECM systému a nejběžnějšími veličinami získávanými a čidly v takovýchto systémech obsazených. Pro komunikaci a čtení dat z elektronických řídících jednotek DDEC se používá aplikace s názvem Detroit Diesel Diagnostic Link (DDDL) kde např. u verze 7.04 má aplikace tyto funkce:

1.Schopnost provádět standartní operace s chybovými kódy
2. přístup ke standartním a rozšířeným řešením krizových situací
3. schopnost čtení aktuálních hodnot veličin
4. nastavení a správa identifikačních parametrů
5. schopnost provádět běžné servisní úkony
6. shopnost přehrát historii záznamu připojených aktivit
7. schopnost automatické aktualizace ze serveru

Aplikace DDDL je instalována na systémech Windows, vyžaduje přístup na server DDC a vyžaduje paralerní port a usb port pro připojení komunikačního kabelu pro komunikaci přes devítipinový konektor na straně ECM (obr. 17).[4]



obr. 17 zapojení 9-pinového kabelu aplikace DDDL

3. INTERNÍ A EXTERNÍ KOMUNIKAČNÍ SYSTÉMY

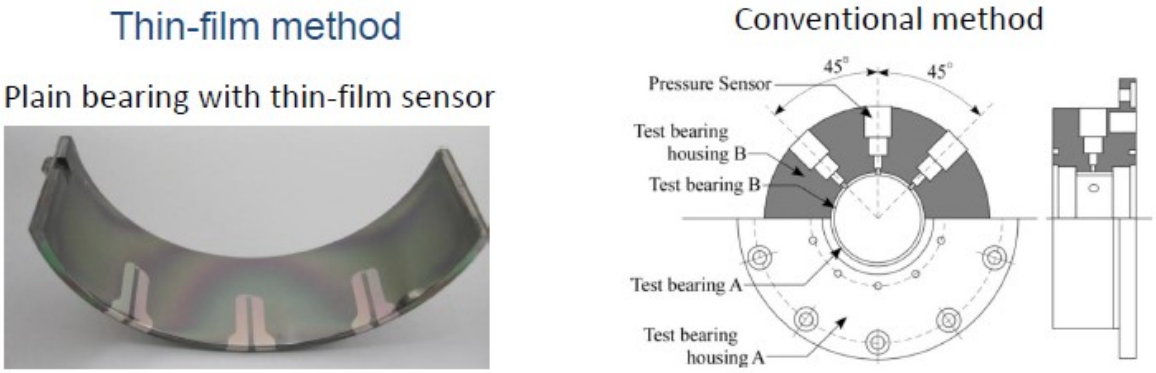
Následně, po rozboru ve srovnání s dnešními multi ECM systémy jednoduchého modelu bakalářská práce přechází k popisu složitějších systémů obsahujících až sto elektronických čidel, modulů a elektronických řídících jednotek. Zvětšující se počet čidel zapřičinil nutnost rozdělení do bloků podle povahy snímaných veličin, přibývá více typů sběrnic. Zvětšuje se také počet čidel umístěných na vnějšku vozidla a interakce vozidla s okolím obecně.

3.1 Nové druhy čidel a jimi snímané veličiny

Rapidní rozvoj osazování automobilů elektronickými čidly, kde průměrný meziroční nárůst se pohybuje v oblasti čtrnácti procent a fakt, že komponenty v automobile jako pohonná jednotka, převodovka, brzdy, odpružení, chladící soustava, výfuková soustava, karoserie a interiér jsou osazeny až desítky čidel každý, sebou přináší mnoho nových druhů čidel, snímaných veličin a událostí, způsobů snímání a využitých technologií. Přibývá velké množství podle norem povinně snímaných veličin převážně v oblasti bezpečnosti a ekologie. Přesto stále zůstává v převaze počet nestandartních prvků, kde nové typy čidel je možno zaznamenat v oblasti zábavy, komunikace a komfortu vozidel.

3.2 Pohonná jednotka

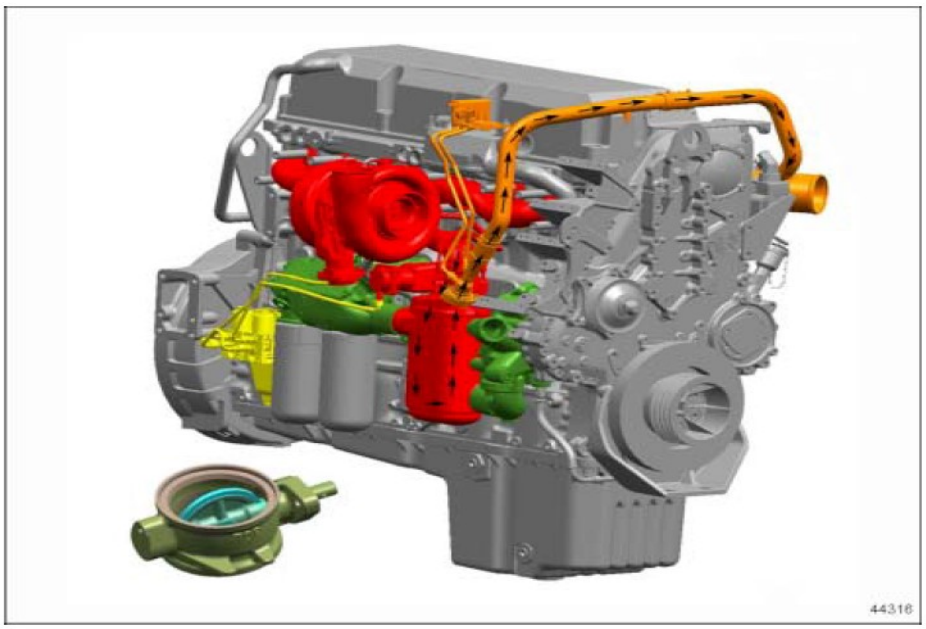
Thin-film sensor (obr. 18) je druh čidla, které je umístěno uvnitř pohonné jednotky na částech vyžadujících monitorování jejich teploty nebo tlaku okolních částí. Podle typu čidla je schopno snímat teplotu nebo tlak a dosahuje přibližně 0.4 mm tloušťky. Nahrazuje tlakové čidla používané v pohonných jednotkách a díky tohoto čidla je možno předejít poškození mechanických částí z důvodu nedostatečné lubrikace a okolních vibrací a tlaku.[1]



obr. 18 Thin film čidlo

Flex Fuel Sensor je čidlo monitorující obsah etanolu v palivovém systému vozidla. Čidlo je umístěno mezi regulátorem tlaku paliva a palivovou nádrží a zde sníma obsah etanolu vracejícího se do palivové nádrže.

Magnetoresistive sensor je čidlo nahrazující indukční čidlo a monitorující rychlost a počet otáček motoru. Díky velké citlivosti na magnetické pole zaručuje snímání otáček motoru s mnohem větší přesností.



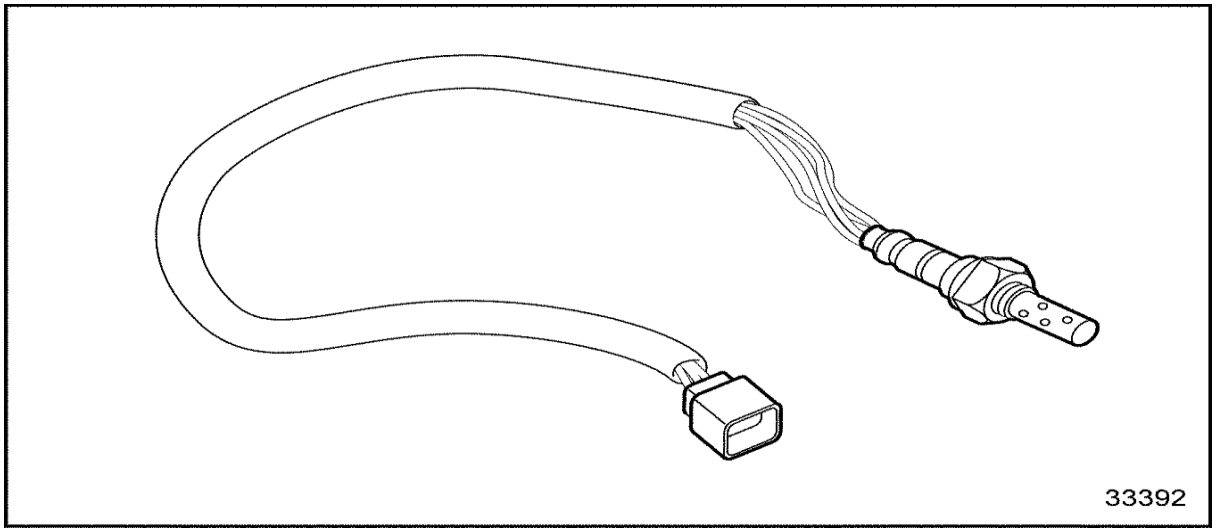
obr. 19 EGR system

Engine knock sensor je čidlo monitorující nežádoucí vibrace a hluk motoru, spojený s rizikem budoucího poškození motoru, nepravidelného chodu a špatného spalování. Toto čidlo je piezo elektrické zařízení, které pod mechanickým zatížením produkuje elektrické napětí.

3.3 Výfukový systém

Delta pressure feedback EGR sensor je čidlo umístěné na redukčním potrubí exhaust gas reduction (EGR) tedy na potrubí regulující výfukové plyny (obr. 19). Proces probíhá ve smyslu řízené regulace výfukových plynů a vrácení nespálených splodin zpět do sání motoru. Čidlo umístěné na EGR čte změny tlaku a porovnává je. Oxygen sensor (obr. 20) tedy čidlo pro monitorování obsahu kyslíku v okolí katalyzátoru přetrvává a jeho počet se znásobil až na čtyři čidla na vozidle.

Ammonia sensor tedy čidlo pro monitorování obsahu amoniaku přispívá ke zlepšení spalování a redukci spalin a obsahu amoniaku ve výfukových plynech.



obr. 20 čidlo kyslíku – oxygen sensor

3.4 Převodovka

Transmission control unit (TCU) tedy jednotka řídící automatickou převodovku se stává nedílnou součástí dnešních automobilů vybavených automatickými převodovkami a figuruje jako komunikační most mezi ECM, čidly a mechanickými částmi uvnitř převodovky. Typickými čidly a moduly spolupracujícími s TCU jsou vehicle speed sensor (VSS) tedy čidlo pro snímání rychlosti vozidla, wheel speed sensor (WSS) čidlo pro snímání otáček kol automobilu, throttle position sensor (TPS) čidlo pro snímání pozice pedálu akcelerace, transmission fluid temperature sensor (TFT) čidlo pro snímání teploty převodové kapaliny uvnitř převodovky, traction control system (TCS) modul kontrolující prokluz a smyk kol vozidla, cruise control module (CCM) modul pro řízení funkce tempomatu.

3.5 Odpružení

LVDT position sensor je čidlo spolu s dalšími jako Magni-tec či rotační a lineární potenciometrická čidla snímající zatížení naprav a pozici tlumičů a zajišťují automatickou regulaci světlé výšky vozidla při různém zatížení a náklonu u pružinových, vzduchových a hydraulických typů odpružení. Na základě informace o náklonu vozidla ECM nastavuje úhel světlometu.

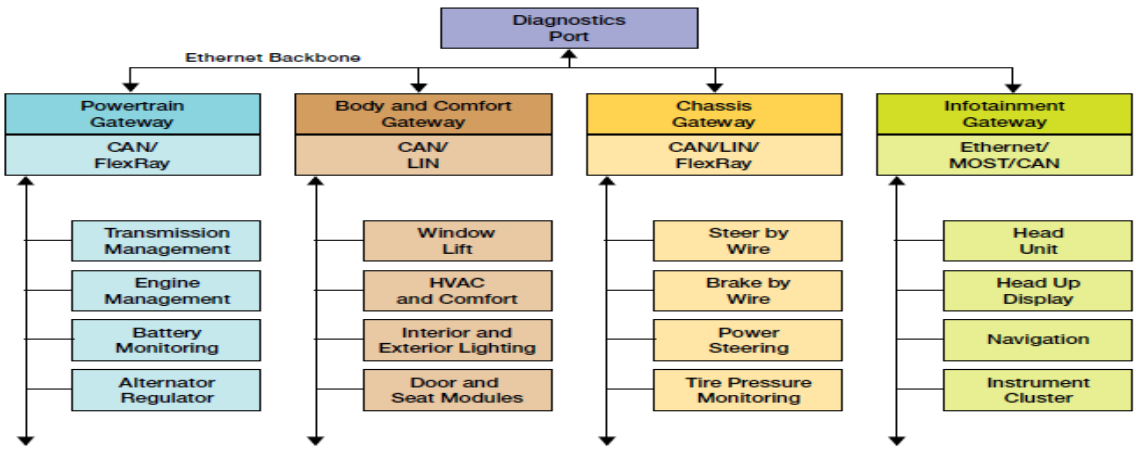
3.6 Interiér

Airbag ECU tedy řídící jednotka pro funkci bezpečnostního systému vozidla spolupracující s bezpečnostními nafukovacími vaky a sadou čidel detekujících náraz je systém, který se neustále rozvijí a v dnešní době obsahují vozidla až deset bezpečnostních vaků a několik čelních, zadních a bočních čidel nárazu.

Typické čidlo nárazu využívá fyzikálních vlastností jevu přetížení, kdy na pohybové ústrojí působí síla, která zapřičiní sepnutí dvou kontaktů, uzavření obvodu a vyslání signálu do ECU pro aktivaci bezpečnostního nafukovacího vaku.[12]

3.7 Dělení čidel do bloků

V moderních elektronických systémech automobilů se tento obrovský počet čidel dělí (obr. 21) do skupin podle účelu: bezpečnostní čidla, diagnostická čidla, čidla pohodlí a komfortu, čidla prostředí.



obr. 21 Domény

Existuje další rozdělení čidel do tzv. domén, kterému koresponduje také několik druhů sběrnic.

3.8 Domény

Powertrain gateway (pohonné ústrojí) – do kterého spadá oblast pohonné jednotky, převodovky, diferenciálních skříní. Tato doména komunikuje s dalšími moduly po sběrnici CAN a Flex Ray.

Body and comfort gateway (karoserie a interiér vozu) – čidla pro bezpečnost a pohodlí uvnitř i vnějšku vozu. Tato doména komunikuje po sběrnicích CAN a LIN.

Chassis gateway (podvozek a nápravy) – čidla pro sledování jízdních vlastností. Tato doména komunikuje po sběrnicích CAN, LIN a Flex Ray.

Infotainment gateway (informační centrum) – moduly pro sledování polohy vozidla a informaci z okolního prostředí. Tato doména komunikuje přes sběrnice Ethernet, CAN a MOST.[2]

3.9 Komunikace s okolím

Existuje několik druhů přenosu informace z vozidel do monitorovacích center či přenosu informace mezi vozidly.

Wireless Access for vehicular environment (Wave) je přenos využívající standart IEEE 802.11p

Communication Access for Land Mobiles (CALM) je typ komunikace využívající rozhraní 2G/3G, infrared, millimeter–wave a mobilní bezdratový přenos.

Zigbee communication standartu a visible light communication (VLC) standartu se také dostává vysoké popularity. [14]

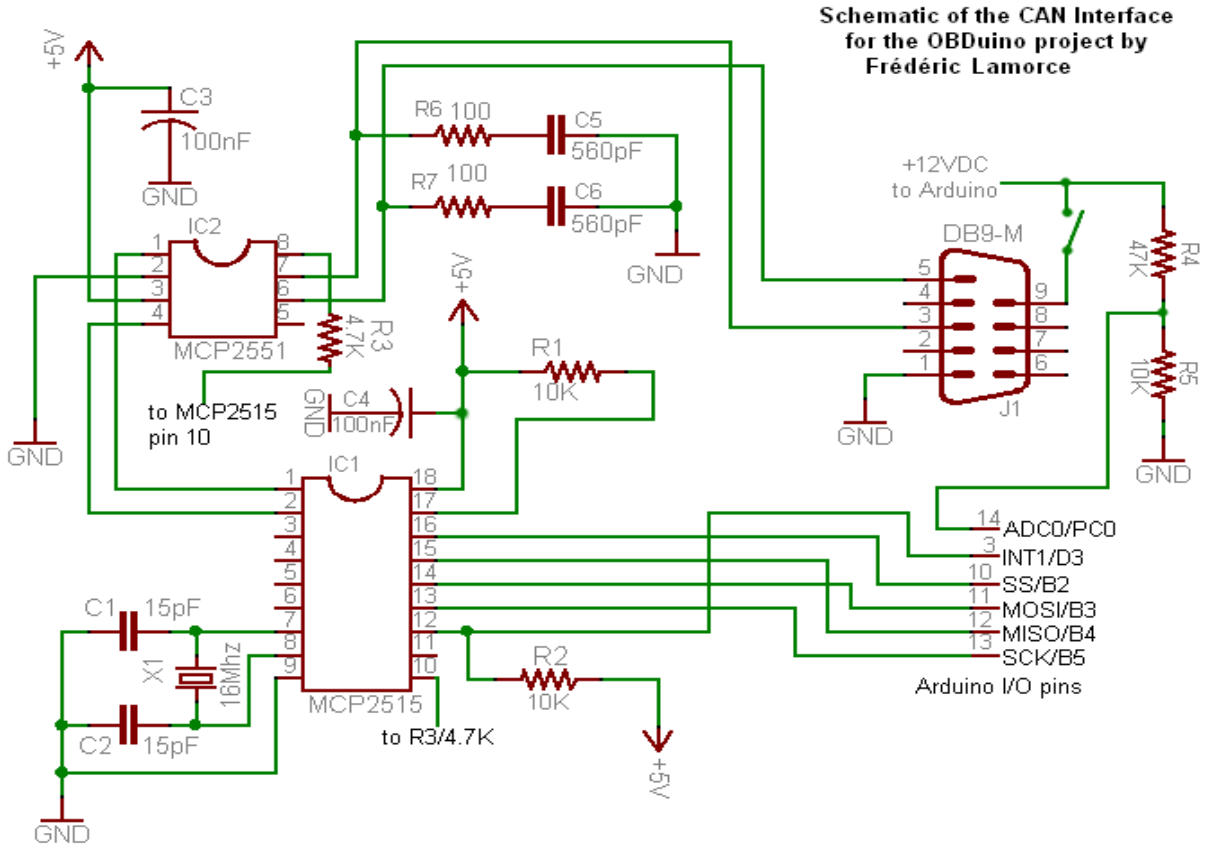
Konkrétním příkladem přenosu dat v oblasti nákladních automobilů, strojních a důlních strojů je využití komunikační jednotky CalAmp. Tato jednotka (obr. 22) využívá schopnosti připojení se na elektronickou řídící jednotku vozidla ECM a umožnuje přenos informace z ní získané přes satelitní a mobilní typ připojení. Sama o sobě obsahuje širokou škálu vestavěných čidel jako např. čidla pohybu, zrychlení, nárazu a také vestavěné funkce využívající GPS navigaci jako vymezení GPS oblasti pohybu vozidla tzv. GPS fence (plot) technologie či AGPS, tedy vysoce přesného zaměření. [6]

obr. 22 CalAmp jednotka

Tato jednotka je jakousi alternativou získávání dat u vozidel s méně sofistikovanými elektronickými systémy a menším počtem čidel. Dalšími oblastmi kde nalezneme prostor pro rozvíjení alternativních řešení získávání hodnot veličin automobilů a jejich další přenos mimo vozidlo jsou studentské projekty, domací projekty, individuálně vyráběné vozy.

3.10 Obduino

Tento projekt je založen na využití jednoduchého mikrokontroleru od firmy Arduino, výrobce open source (volne šiřitelný) freeware a hardware. Pracuje na principu velice podobném aplikacím vozů standartní produkce. Koresponduje s rozhraním OBD a rozpozná část parametrů identifikace PID. Základními údaji získávanými v tomto projektu (obr. 23) jsou okamžitá spotřeba paliva, rychlost, otáčky motoru, teploty kapalin a hodnoty elektrického napětí. [18]



obr. 23 Obduino project

Na základě těchto veličin Arduino mikrokontroler (obr. 24) propočte a zobrazí na připojené obrazovce údaje o průměrné spotřebě, spotřebě paliva od začátku cesty, dojezd, spotřebované palivo při volnobežném chodu pohonné jednotky. Chybové kódy nejsou v tomto projektu zobrazovány.

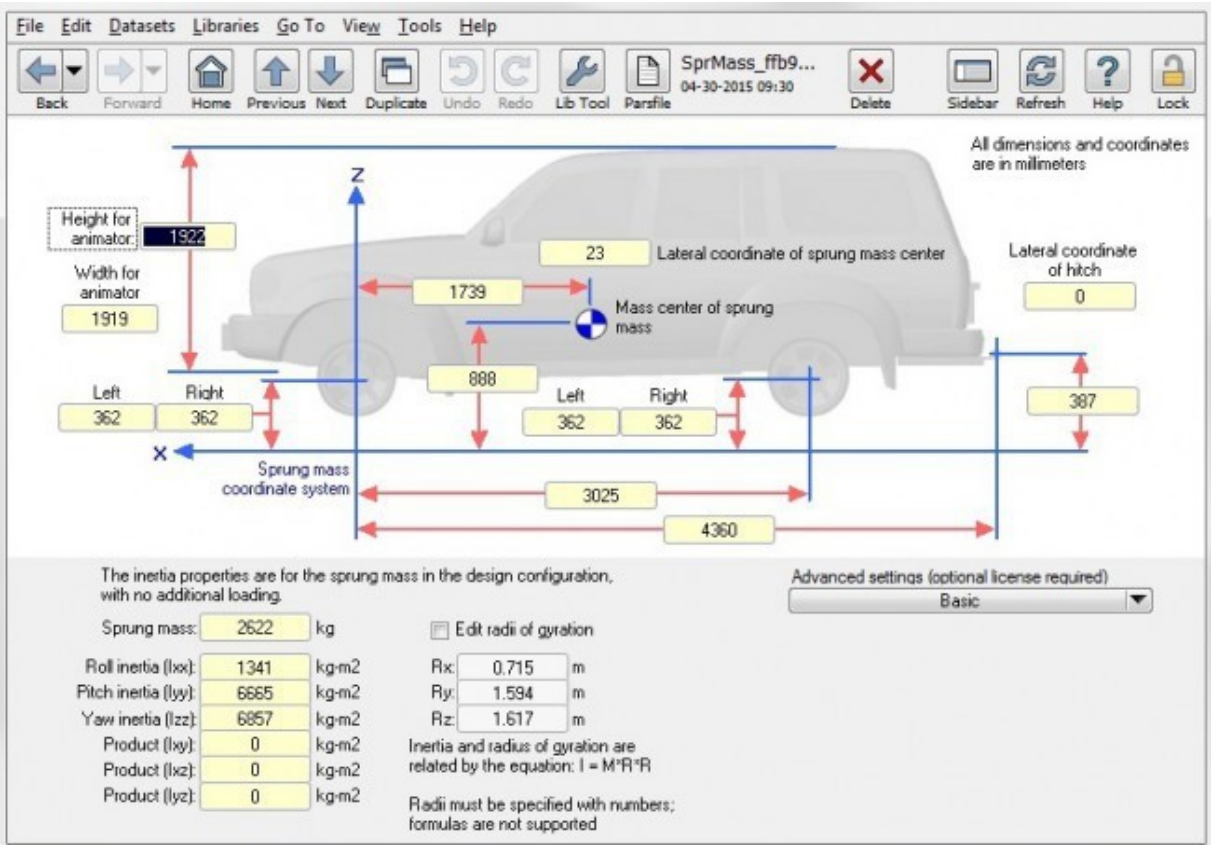
obr. 24 Arduino v projektu OBD

3.11 Carputer

Název pro elektronický kontroler uvnitř automobilu. Tento termín se používá u menších projektů, kde automobil, prototyp vozidla, historické vozidlo, neobsahuje elektronickou řídící jednotku, nebo obsahuje, ale není v projektu využita pro komunikaci. Jedná se o studentské projekty, domací projekty, kde neexituje sjednocené komunikační rozhraní, ani žádné standartní kódy, rychlosti přenosu nebo velikost napájení. Účel a využití se liší individuálně podle projektu a často se funkce carputeru omezuje pouze na základní funkce navigace a infotainment centra. Velmi populární jsou pro tyto typy projektů raspberry a arduino kontrolery. [7]

3.12 Virtuální dyno technologie

Dalším typem projektů bez využití komunikace s ECM, nebo s částečnou komunikací s ECM jsou projekty soustřeďující se speciálně na výkon, dynamiku a jízdní vlastnosti vozu. Pro tyto druhy projektů se používá virtuálních studií, které obsahují simulaci situace (obr. 25) na virtuálním modelu ve virtuálních podmínkách, nebo reálných instrumentů umístěných na fyzickém modelu, kde získaná data jsou zpracována v aplikacích pro výpočet dynamiky.



obr. 25 Nastavení aplikace pro simulaci stability vozidla

Mezi vedoucí firmy nabízející řešení pro individuální požadavky zákazníků patří Mechanical Solutions, MSC software nebo Tesis DYNAware.

V České republice výrobce nákladních automobilů Tatra provozuje testovací areál (obr. 27) vybavený testovacím okruhem a laboratořemi, kde instrumentální čidla fyzicky umístěná na individuálních částech automobilu snímají potřebná data a přenášejí je do počítačové aplikace, která tyto data zpracovává.

CARSIM je aplikace společnosti Mechanical Solutions, dodávaná je ve čtyřech verzích suspSIM, bikeSIM, carSIM, truckSIM s rozšířením externími programy LABview a MATLAB/simulink. [8]

ADAMS je řešení společnosti MSC software a nabízí simulační aplikace využívané pro testování částí již při výrobě. Hlavní druhy testů jsou test durability a vibrací, z kterých je možno odvodit životnost komponenty a dále také integrační test. S rozšířením extérními programy MATLAB nebo Easy5.[9]

DYNA4 framework je produkt od společnosti TESIS DYNAware. Ve svých verzích nabízí řešení pro konkrétní situace.[10]

1. Advanced powertrain – verze pro optimalizování spotřeby vyvíjených prototypů elektrických či hybridních a tradičních vozidel.
2. Engine Professional – verze pro vývoj konceptů bloku pohonných jednotek všech druhů : benzínové, naftové, alternativní paliva.

Tatra trucks a.s. (obr. 26) je naopak příkladem společnosti využívající reálných instrumentů instalovaných na fyzických komponentech prototypů nákladních a osobních vozů (externě), kde při testování dochází k záznamu dat do úložiště v automobilu. [11]

obr. 26 Tatra nákladní automobily

Tyto data jsou později analyzováná specialisty v příslušných laboratořích. Statické testy probíhají přímo v laboratořích společnosti, kdy jednotlivé komponenty jsou vystavovány stresu v podobě např. vibrací nebo nárazu a příslušné čidla snímají data, která jsou později vyhodnocena počítačovými aplikacemi.

obr. 27 Testovací areál Tatra Kopřivnice

4. SPECIFIKACE ZADÁNÍ

Dosud popsaný způsob získávání a přenosu dat je světovým standartem využívaný mezinárodními korporacemi globálních rozměrů. Sledovat aktuální data o vozidle v reálném čase je dnes u mnohých institucí a společností nutnost. Vědět kde se vozidlo pohybuje, je samozřejmost u dispečerů dopravních firem, záchranných a bezpečnostních složek. U těchto typů monitorování centrálu většinou zajímá především poloha vozidla. Majitelé dopravních firem, kteří fyzicky také vlastní svá vozidla mají často zájem kromě polohy vozidla sledovat i další informace o vozidle, převážně data čitelná z ECM.

Avšak v oblasti menších a středně velkých automobilových projektů se otevírá prostor pro vývoj specifické aplikace na přenos dat. Využití zde najde kompaktní řešení, dostatečně rychlé, bez nutnosti registrace a paušálních poplatku.

Aplikace, která je lehce přizpůsobitelná dané situaci, avšak schopná získat, uložit, přenést, analyzovat a zobrazit žádaná data z částí testovaného automobilu uživateli. Existuje mnoho neúplných řešení, které se soustředí jen na konkrétní část práce s daty.

V situacích kdy například studenti, restaurátoři historických vozidel, či vývojáři automobilů na zakázku individuální produkce potřebují monitorovat svá vozidla často vystačí kompaktní řešení. Zjednodušená komunikace přímo s potřebnými čidly na vozidle bez nutnosti přizpůsobení se standartu komunikace s ECM, je pro tyto typy projektu ideální a výše zmínění jsou potenciální cílovou skupinou uživatelů aplikace.

Z důvodu nepravidelného využívání aplikace při testování vozidel, či například studentských závodech, je tato lehká verze komunikace rozumné a ekonomické řešení.

Bakalářská práce je inspirována mimojiné modelem nepřímého čtení dat bez potřeby komunikace se standartním ECM a také potřebami studentských projektů katedry elektrotechniky. V této situaci se nabízí prostor pro vývoj alternativního (zjednodušeného) typu ECM a zjednodušeného typu komunikace, analýzy dat a jejich zobrazení ve webové aplikaci jak v desktopové tak mobilní podobě. Výsledná aplikace najde uplatnění i v případě přenosu dat ze sady čidel instalovaných na VŠB prototypu vozidla Kaipan (obr. 28) s pohonem elektrickou pohonnou jednotkou.

obr. 28 vozidlo Kaipan

5. ANALÝZA

V kapitole analýza je obsažen rozbor informací potřebných ke správnému návrhu implementace aplikace, zahrnující vstupy, výstupy, funkce a okolí aplikace, lineární zápis entit a jejich vztahů, minispecifikace a příslušné diagramy.

5.1 Vstupy

uživatel
vozidlo
událost
záznam
data přenesená z čidel
data kontroly a nastavení

Uživatel
identifikační data o uživatelích systému
userID, jméno, institut, druh uživatele(administrátor, divák, jezdec, analytik),heslo

Vozidlo
data o vozidle, které obsahuje sadu čidel a účastní se testování, tréninku nebo závodu
vehicleID, typ, výkon, druh jednotky

Událost
data o událostech (test, trénink, závod) s vozidlem prováděných
eventID, start, konec, typ akce, časová známka

Záznam
nahrávané hodnoty veličin ve spojení s časem, datumem, místem, druhem snímače
datum, gps_coords, výška, rychlost, otáčky, teplota, tlak, zrychlení, vlhkost, hustota, druh čidla, časová známka

Data přenesená z čidel
data neukládaná, pomocná pouze pro chod aplikace (chybové kódy, počet družic GPS)

Data kontroly a nastavení
data ve formě hodnot a instrukcí načtená jako vstupy z klávesnice

5.2 Výstupy

data uložená v souborech CSV
data uložená v souborech PDF
data uložená v databázi MySQL
data zobrazovaná v mapových podkladech Google a grafech Fusioncharts

Data uložená v souborech CSV
jedná se o strukturovaná data, roztříděná podle datumu a času ve formátu datového řetězce
typu identifikačních hodnot vozidla, čidla, události, jezdce
typu hodnot snímaných veličin

Data uložená v souborech PDF
jedná se o data specifického časového úseku
zobrazena v grafových podkladech FusionCharts

Data uložená v databázi MySQL

jedná se o organizovaná data, uložená v příslušných polích datových entit
typu identifikačních hodnot vozidla, čidla, události, jezdce
typu hodnot snímaných veličin

Data zobrazovaná v mapových podkladech Google a grafech Fusioncharts

vizualizační data ve formě pozice v mapě, identifikace vozidla
křivky v závislosti na zvolených osách, porovnání, limity

5.3 Funkce

validace a formátování
organizace a uložení
transportní řetězec
výpočetní operace
zobrazení dat

Validace a formátování

funkce užité v okamžiku načtení hodnot snímaných veličin, či vstupu z klávesnice
zajišťující správný formát dat, nenulové hodnoty, desetinná místa, správný datum a také vytvářející strukturu datových řetězců pro data formátu JSON

Organizace a uložení

funkce zodpovědné za správné roztřídění načtených hodnot snímaných veličin, vytvoření adresářové struktury a souborů CSV a vkládání datových řetězců do vytvořených souborů

Transportní řetězec

funkce zajišťující správnou návaznost datových toků od okamžiku komunikace s daným čidlem až po uložení dat do souboru CSV, databáze MySQL a zobrazení uživateli

Výpočetní operace

funkce zpracovávající načtená data a provádějící matematické operace ve smyslu
analýzy dat pro rozbor načtených hodnot
doplnění statistik pro vytvoření informativních výstupů
propočtu prediktivních hodnot a tvorbu odhadů
kontroly aktuálních hodnot přímo v reálném čase porovnáváním
zobrazení

5.4 Okolí aplikace

Aplikační okolí obsahuje uživatelé a další návazné systémy s kterými aplikace spolupracuje

Uživatelé

Administrátor – osoba s rozšířeným oprávěním a zodpovědností za nastavení aplikace, vkládání a mazání položek aplikace, mající také standartní přístup a privilegia

Divák – osoba se základní úrovní přístupu bez nutnosti registrace, mající pouze observační typ privilegia

Jezdec - osoba se základní úrovní přístupu s nutnosti registrace, mající observační a nastavující typ privilegia

Analytik - osoba se základní úrovní přístupu s nutnosti registrace, mající observační a nastavující typ privilegia a možnost organizace a práce s daty

Další návazné systémy

Satelitní navigační systém GPS

5.5 Lineární zápis entit a jejich vztahů

Entity

Uživatel
- user (userID, jméno, institut, druh uživatele(administrátor, divák, jezdec, analytik),heslo)

Vozidlo
- vehicle (vehicleID, typ, výkon, druh jednotky)

Událost
- event (eventID, start, konec, typ akce, časová známka)

Záznam (nahrávky)
- recordings (datum, gps_coords, výška, rychlost, otáčky, teplota, tlak, zrychlení, vlhkost, hustota, druh čidla, časová známka)

Vztahy

has_Events(recordings, events) N:1
has_Users(recordings, users) N:1
has_Vehicles(recordings, vehicles) N:1
has_Events_Vehicles(events, vehicles) N:N
has_Users_Events(users, events) N:N

5.6 Minispecifikace - zapsání nové události

1. Zobrazení formuláře pro zápis nové události
2. Vložení údajů uživatelem
3. Kontrola vložených dat
4. Uložení záznamu do tabulky Recordings
5. Přechod do záhlaví strany

5.7 Diagramy

Diagram případu užití (use case)

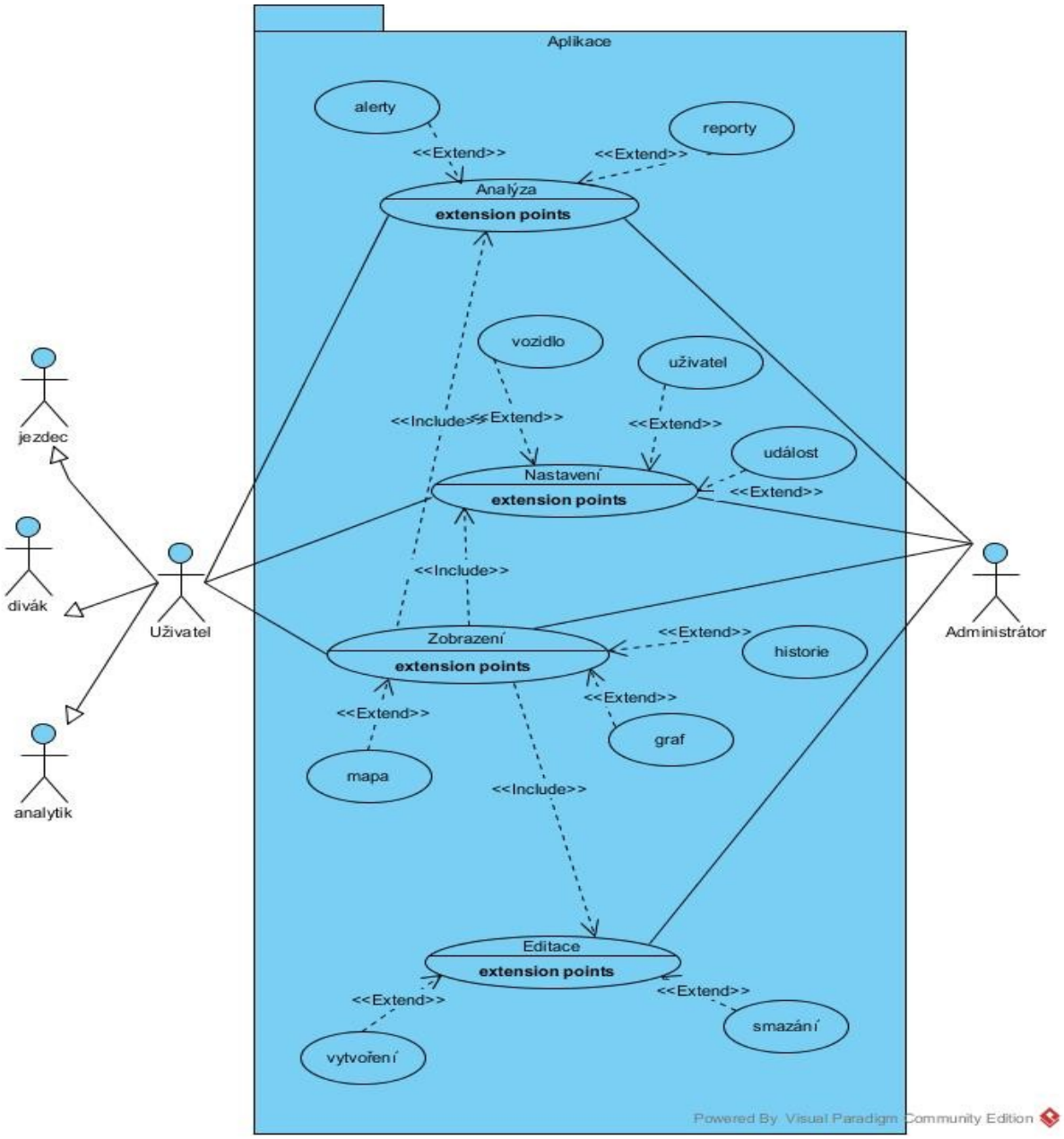
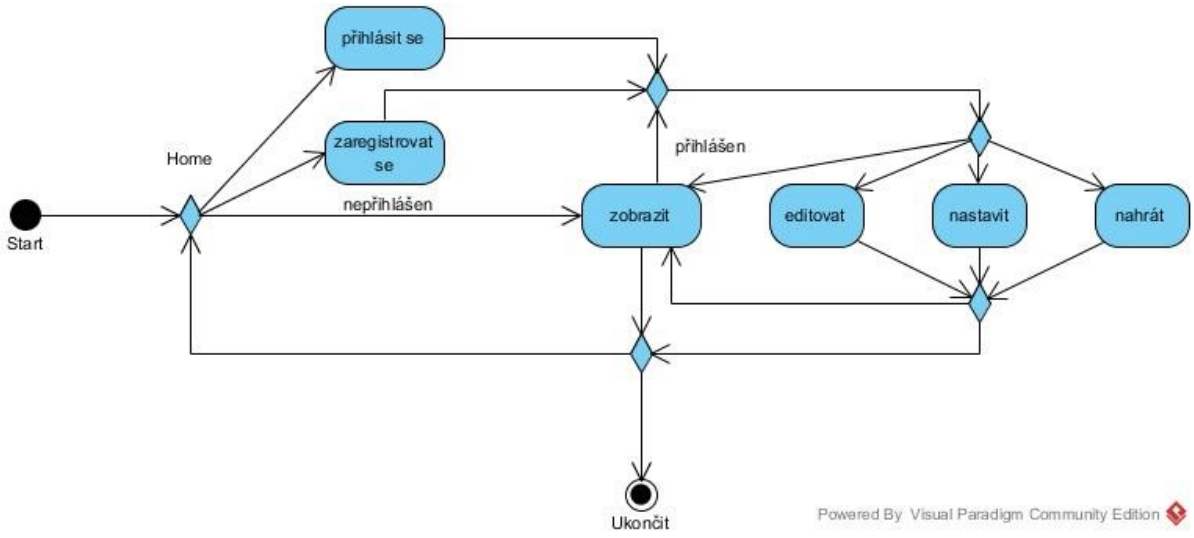
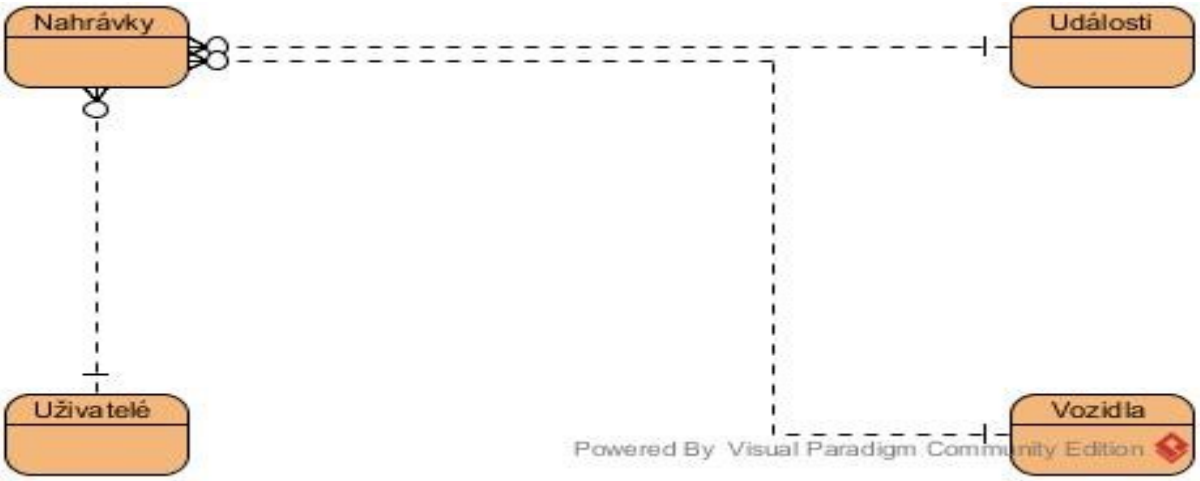


Diagram aktivit (activity)



ER diagram(entity relationship)



Sekvenční diagram(sequence)

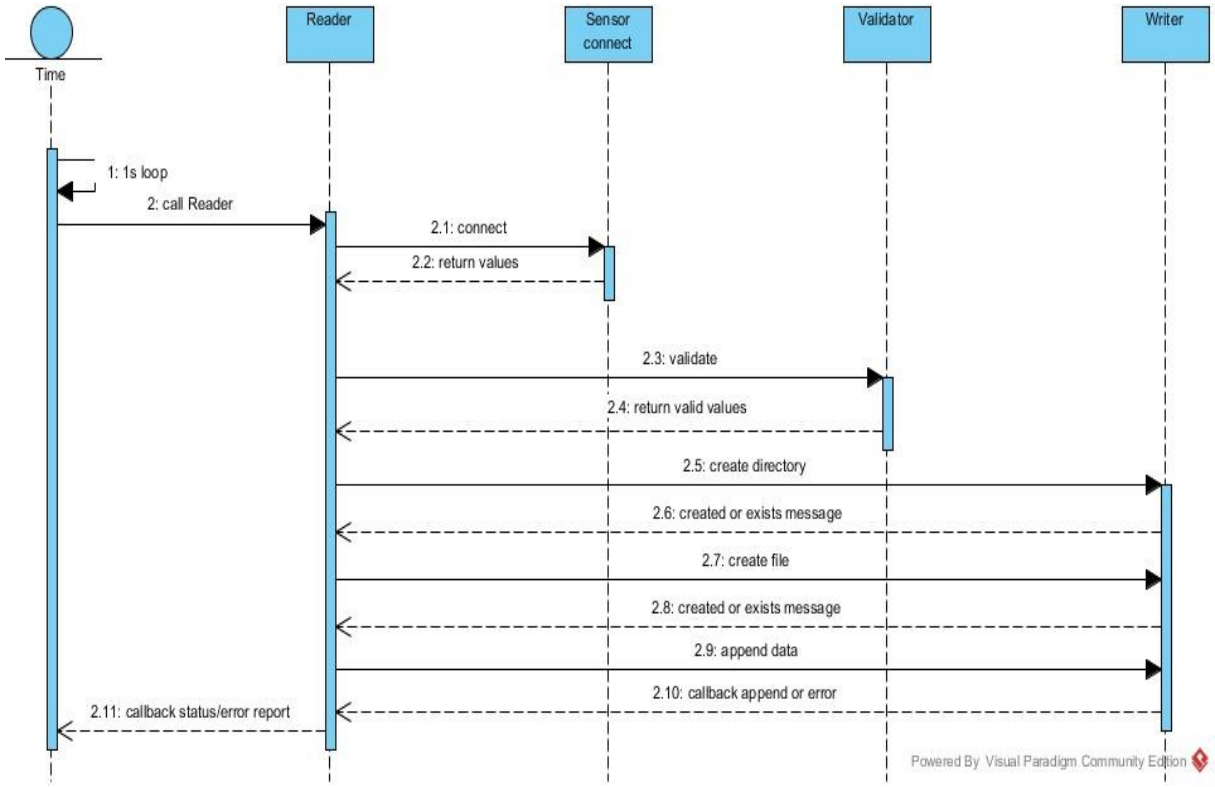


Diagram komponent(component)

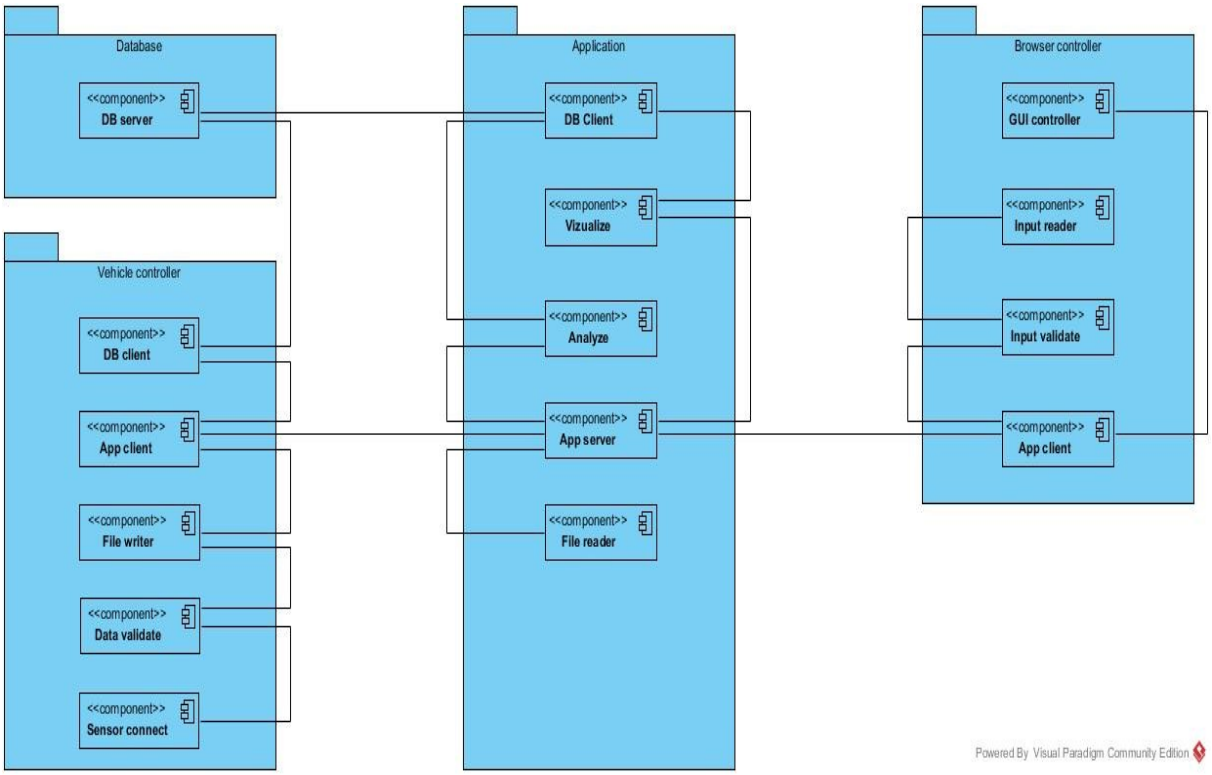
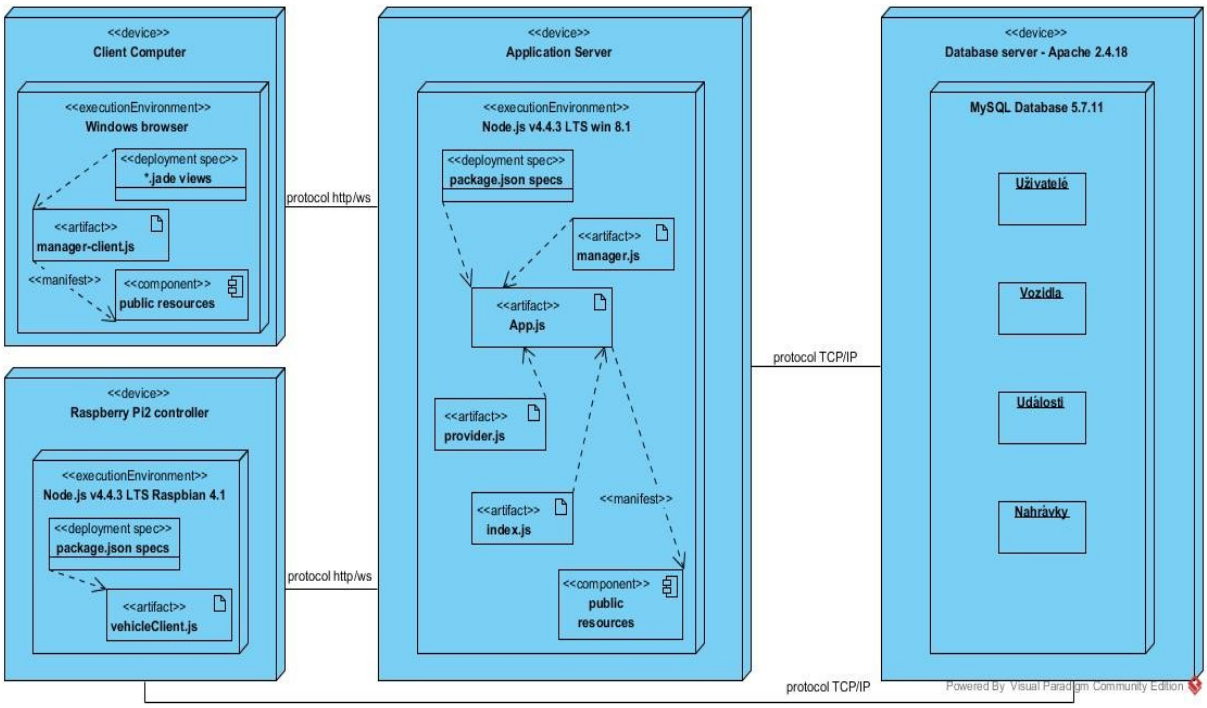


Diagram nasazení(deployment)



6. NÁVRH

Při návrhu aplikace je možno rozdělit celkovou oblast problému do několika konkrétních skupin a na každou z nich navrhnout část aplikace, která danou část problému řeší. Každá část aplikace bude obsahovat několik komponent zabývajících se specifickým řešením. Každá část bude také využívat různých technologií, které je nutno zmínit a vysvětlit důvod jejich použití. Koncové komponenty je také nutno propojit tak, aby výsledná konfigurace dosáhla správné funkčnosti.

Původní navržená verze aplikace využívá technologie scriptovacího jazyka PHP, ale po hlubší rozvaze a analýze problému je nahrazena novější a vcelku populární technologií programového prostředí pro vývoj převážně serverových aplikací Node.js, framework postavený na skriptovacím jazyce javascript. Node.js se jeví jako zcela ideální z důvodu globální dostupnosti ve všech oblastech které aplikace řeší a to především díky množství open source (volně šiřitelných) modulů. Hlavní předností je však jednoduchá událostmi řízená architektura, schopna asynchronních vstupně výstupních operací, čímž se Node.js stal optimální volbou pro webové aplikace zpracující data v reálném čase a také hry. Díky těmto vlastnostem je také samozřejmě ideální volbou pro vývoj aplikace bakalářské práce.

6.1 Část napojení aplikace na měřící zařízení a čtení dat

Tato část je okrajovou částí aplikace a běží externě v odděleném operačním systému, na odděleném hardwarovém zařízení umístěném spolu s čidly na sledovaném vozidle. Při její obměně nedojde k zásadní změně funkčnosti celé aplikace. Při navrhovaném využití kontroleru Raspberry Pi2 je využit s Raspberry standartně dodávaný operační systém Raspbian ve kterém je spuštěn Node.js a komponenta vehicleClient obsahující další callback funkce.

Předvytvořený modul je využit pro komunikaci, tedy čtení hodnot a ovládání čidel. Čidla jsou ke kontroleru připojená přes tzv. GPIO porty.

V případě nutnosti záměny kontroleru Raspberry za standartní laptop s nainstalovaným Node.js je možno rychlé adaptace ve formě změny modulu, v případě připojení čidla bezprostředně do sériového portu laptopu.

6.2 Část validace a konverze dat

Tato část je standartní vstupní branou dat do aplikace. Je nezbytnou součástí nezávislou na způsobu získávání dat a typu hardwarového nástroje. Ať už při zapojení kontroleru Raspberry Pi2 nebo standartního laptopu, data je nutno následně validovat a uzpůsobit pro další přenos. Také běží na externím hardwarovém zařízení na sledovaném vozidle. Callback funkce pro validaci dat této části porovnávají a parsují typy proměnných v závislosti na ukládaných hodnotách. Další validační funkce jsou využity při načítaní vstupů z klávesnice, např. při zadávání datumu v části pro ukládání hodnot.

6.3 Část přenosu dat

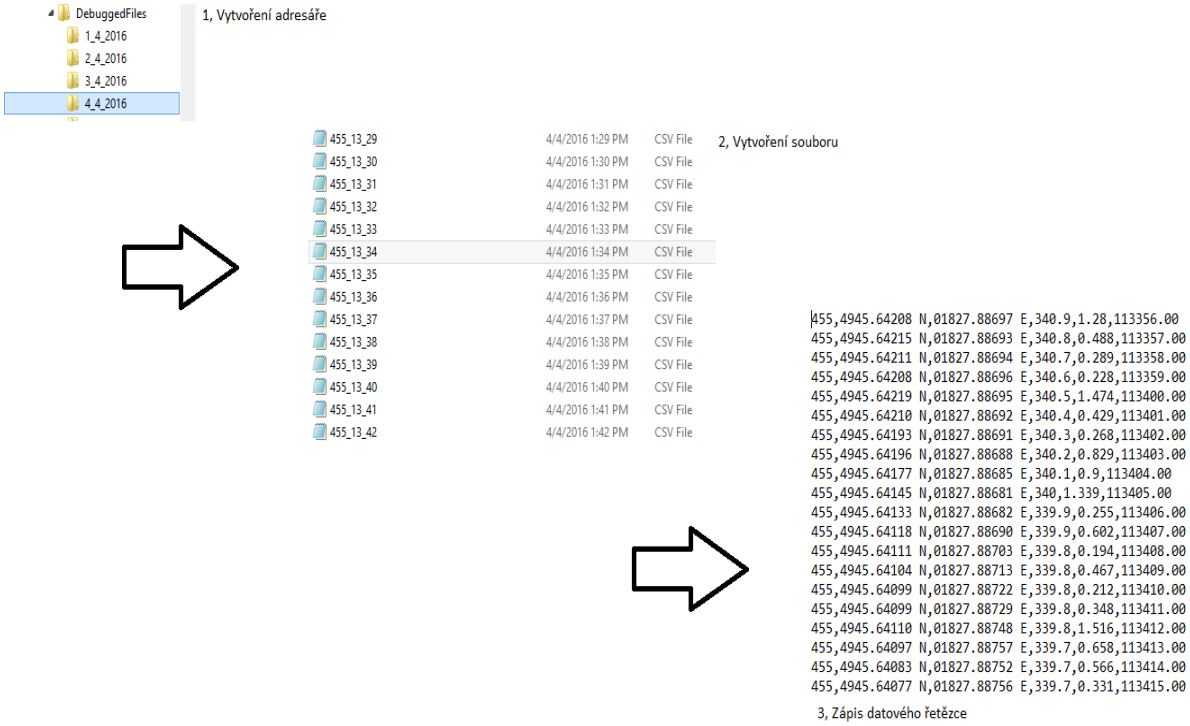
Pro přenos dat a zpětnou komunikaci s čidly je nutno napojení samotné komponenty clientVehicle k hlavní části aplikace app.js. Hlavní část aplikace, neboli jakési rozcestí, u jiných programovacích jazyků jako C, či C++ převážně nazvané „main“ běží odděleně na serverové části aplikace.

Pro komunikaci protokolem TCP je využit modul NET přenášející data pomocí metody write v momentě obdržení dat monitorované metodou on.data. Komunikace probíhá obousměrně pomocí sockets a je možné ji v kterémkoliv momentě ukončit oběmi stranami pomocí on.end události. Na rozdíl od protokolu WS(websockets) operujícím nad protokolem HTTP je TCP socket řetězcově orientovaný. Při zápisu dat do databáze funkce filetoDatabase využívá standartní komunikace se serverem obsahující MySQL.

6.4 Část uložení a správy dat

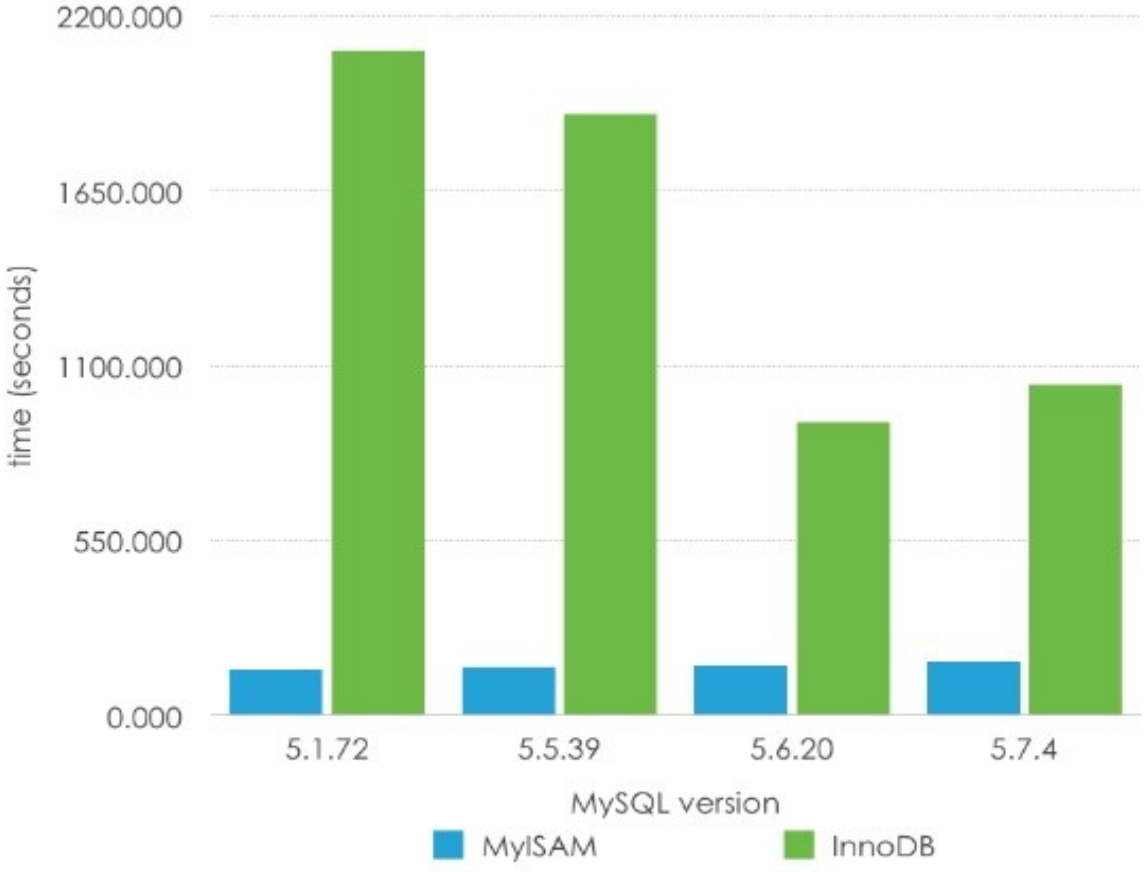
Ukládání dat probíhá ve dvou formách. Forma prvního typu uložení je zapsání získaných dat do souboru na datové úložiště hardwarového zařízení nacházejícím se přímo na vozidle. Druhá forma uložení dat je zapsání dat do databáze externího serveru obsahující MySQL. V obou případech impuls k uložení vychází z části aplikace clientVehicle.

Při ukládání do souboru CSV již validovaná data organizovaná v datovém řetězci odděleném čárkami jsou ukládána standartně každou sekundu, či v jiném požadovaném časovém intervalu. Pomocí modulu FS (file system) pro správu souborů a adresářů je vytvořen adresář pro zápis hodnot daného dne, do kterého je každou minutu zapisován nový soubor s identifikačním číslem vozidla a časem vytvoření souboru (obr. 29). Před uložením je do tohoto souboru každou sekundu připojován datový řetězec s novými daty.



obr. 29 práce se systémem souborů

Při vytvoření nového souboru zároveň probíhá jednorázový zápis souboru do databáze způsobem „LOAD DATA LOCAL INFILE“ . Pro tyto účely byl nastaven typ databáze MyISAM jež dosahoval v testech výsledků 12x rychlejšího přístupu (tab. 4).



MySQL Version	5.1.72	5.5.39	5.6.20	5.7.4
MyISAM	141.414	149.612	155.181	166.836
InnoDB	2091.617	1890.972	920.615	1041.702

tab. 4 srovnání MyISAM a InnoDB

Protože architektura node.js je postavená na využití asynchronních operacích, samotné ukládání tak nebrzdí chod aplikace, přestože není využito multi vláknového vykonávání datových a instrukčních toků. Je třeba se ale zamyslet na návaznosti dalších operací právě z důvodu asynchronizace, protože jednoduše nebudeme vědět a nemáme zaručeno kdy bude ukládání dokončeno. Pro tyto typy situací použijeme další callback funkce, které jednoduše vyčkají na dokončení předchozí operace a spustí se následně až po zavolání.[28]

6.5 Část serveru aplikace

Aplikace poběží na hardwarovém zařízení serveru spravovaném operačním systémem Windows s frameworkem Node.js. App.js je základním kódem, který provede komunikaci s dalšími externími a lokálními částmi kódu aplikace. Lokální kódy umístěné a vykonávané v serverovém prostředí a externí kódy umístěné v externím klientském prostředí vytvářejí MVC strukturu, tedy MODEL-VIEW-CONTROLER architekturu, která zajistí oddělení úrovní dat. App.js obsáhne základní funkce pro připojení klienta vozidla a klienta prohlížeče pomocí TCP protokolu a sockets modulu net, HTTP protokolu a websockets, korespondujících programovacích modulů a socket.io frameworku. Tento framework řeší navázání spojení transparentně, tedy automaticky změní typ protokolu přenosu je-li to možné z HTTP na WS.

V části controler bude umístěna část kódu pro komunikaci s databází, která řeší přístup k datům na databázovému serveru pomocí standartních query dotazů jazyka MySQL. Zde bude také místo pro veškeré komponety výpočetních operací určených k analýze dat pro rozbor načtených hodnot, doplnění statistik pro vytvoření informativních výstupů, propočtu prediktivních hodnot a tvorbu odhadů, kontroly aktuálních hodnot přímo v reálném čase porovnáváním. V části model bude zajištěna programová reprezentace entit pomocí korespondujících proměnných. Mezi lokální kódy serverového prostředí patří také ty obsahující funkce spojené s administrativou, jako například login validace a další funkce pro nastavení ke kterým je umožněn přístup pouze administrátorovi aplikace.

6.6 Část klienta a zobrazení dat

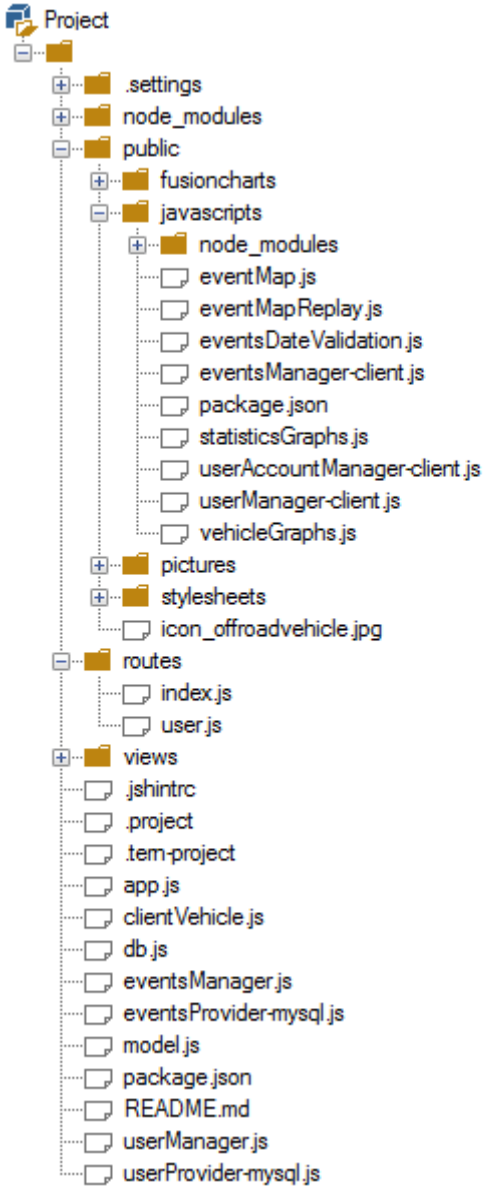
Na hlavní část aplikace app.js dále naváže klientská konzole, která zajistí zobrazení dat, pomocí které se uživatel připojí a komunikuje s hlavní částí. Tato část kódu poběží na klientském hardwarovém zařízení, které může mít podobu desktopového počítače, přenosného laptopu a dalších mobilních zařízení včetně telefonů. Programovacím jazykem této části aplikace je standartní javascript, který je základem node.js, tudíž je naprosto kompatibilní. Javascript kód *manager-client.js je zpracován prohlížečem (browser) daného hardwarového zařízení klienta, zajistí obousměrné spojení pomocí protokolu websockets a zobrazí data v graphic user interface (GUI). GUI je grafické rozhraní uživatele a mělo by být adaptabilní na různé velikosti obrazovek hardwarových zařízení klienta. Aplikace pro tyto účely použije tzv. Bootstrap framework ve spolupráci s Bootswatch šablonami, což je nářadí určené k tvorbě responsivního formátu webových stran. Responsivního, tedy jednoduše schopného se přizpůsobit různým velikostem obrazovek. Tento framework má vestavěné metody pro automatické přeorganizování části webové strany podle velikosti obrazovky, popřípadě zvětšení či zmenšení daných komponent webové strany. Programovací jazyk JADE je využit k dosažení čitelnějšího HTML kódu bez nutnosti opakování programových bloků. Výsledná webová strana je takto připravena zobrazit data. Uložená data je možno následně uživateli prezentovat ve vizuální podobě v několika formách. Pro zobrazení dat v tabulkách využije aplikace šablon obsažených v nářadích Bootstrap a Bootswatch, pro zobrazení dat v mapových podkladech využije aplikace nářadí google knihoven pro práci s mapou Google API a práci s navigací geolocation API. Vestavěné funkce zajistí např. zobrazení aktuální polohy, vzdáleností a trasy. Pro zobrazení dat v grafech aplikace využije knihoven FusionCharts zaměřené na hluboce propracované vizualizace grafových komponent a vizualizace real time dat, tedy dat v reálném čase. Fusioncharts je profesionální nářadí s nutností registrace a poplatku, nabízí však časově neomezenou zkušební verzi, která vygeneruje ve výsledné grafové komponentě v rohu logo společnosti. [27]

7. IMPLEMENTACE

Následným krokem je programování aplikace, které si popíšeme v této kapitole včetně její struktury, daných komponent, konektorů a jejich konfigurace. Také způsob práce, přípravy na testování a povahu vstupních a výstupních dat. Při již skutečném programování vychází projekt aplikace z předvytvořené šablony běžného základního projektu node.js (obr.30) s Express frameworkem.[21][22][23]

7.1 Struktura projektu

Vlastní programování potom obsahuje všechny očekávané programové části M – V - C architektury, moderní jade šablony pro větší přehlednost a snadnější práci a také použití dalších frameworků a modulů, jejichž funkčnost a vzájemné propojení si nyní popíšeme.



obr. 30 struktura projektu

Projekt je iteračně vyvíjen a zároveň testován ve studijním prostředí a je nereálné popisovat zde změny každé iterace zvlášť, ale přesto těsně před dokončením posledních úprav a následným testováním aplikace ve skutečném terénu si v další kapitole popíšeme průběh konečného testování a poslední změny. Aktuální třicátá šestá verze aplikace obsahuje tři tisíce tři sta osmnáct řádek programovaného kódu nepočítaje podpůrných modulů a vygenerovaných kódů. Velikost by se již neměla podstatně měnit, protože následné testování se bude zabývat převážně laděním chyb a zajišťováním výjimek.

V základním adresáři projektu popíši tyto podadresáře:

- node_modules
- public
- routes
- views

a tyto soubory:

- clientVehicle.js
- app.js
- package.json
- model.js
- eventsManager.js
- eventsProvider-mysql.js
- userManager.js
- userProvider-mysql.js
- db.js

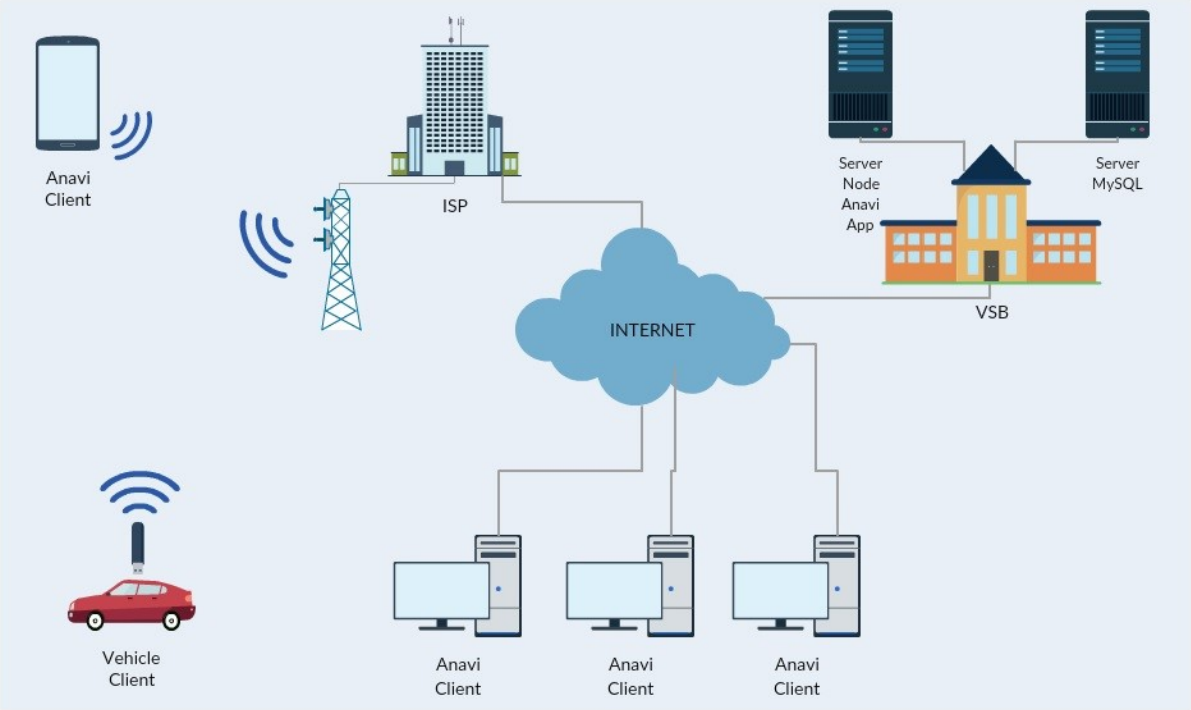
Adresář *node_modules* obsahuje veškeré moduly se kterými celková aplikace pracuje. Jejich inicializace probíhá načtením souboru package.json, který obsahuje jméno modulu a jeho verzi. Pro získávání modulů a jejich aktualizaci používá node.js utilitu npm.

Adresář *public* obsahuje veškeré podpůrné kódy jako scripty, ilustrace, styly nebo třeba ikonu aplikace icon_offroadvehicle.jpg. Při popisu kódů javascript se do něj vrátíme.

Adresář *routes* obsahuje ztěžejní část, kód index.js, který slouží jako směrovací tělo víceúrovňové architektury node.js založené na tzv. exportech. Více při popisu souboru index.js.

Adresář *views* obsahuje všechny kódy upravující vzhled a funkčnost grafického prostředí aplikace. Tento druh souboru s příponou .jade využíva scriptovacího jazyka jade pro vytvoření šablon a jednoduššího a méně obsáhlejšího kódu. Soubory následně pracují s javascripty, které nacházející se v klientském prostředí pak komunikují s hlavním kódem umístěným na serveru aplikace.

Pro získání lepší představy o rozmístění kódů a funčnosti celé aplikace je zde uveden obrázek č. 31. V budově VŠB je na školním počítači spuštěn kód app.js. Ten dále spolupracuje hlavně s kódy index.js a manager.js a provider-mysql.js. Na klientu vozidlo je spuštěn kód clientVehicle.js který se přes TCP protokol připojí na server aplikace spuštěný kódem app.js, inicializuje sensory a začne posílat data. Všichni ostatní klienti pak využívají běžného internetového prohlížeče a pomocí protokolu http zadají request a je jim načten jeden z korespondujících kódů z adresáře views a javascripts. Díky websockets s pomocí podpůrného frameworku socket.io jsou následně reálná data aktualizována na obrazovce klienta. Zároveň jsou data ukládána do databáze umístěné na odděleném serveru a ve stejném čase také do souboru na paměťovou kartu přímo v mikrokontroleru na vozidle.



obr. 31 aplikační schema

Na klientu provozovaném na vozidle není potřeba obrazovky pro zobrazování, internetového prohlížeče ani nutnosti znát a mít veřejnou IP adresu, což byl požadavek pro správnou funkcionalitu aplikace. Proto je použit klient u kterého se IP adresa může dynamicky měnit, TCP protokol pro přenesení aktuálních hodnot sensorů a připojení přes mobilní síť operátora, v našem případě Tmobile.

Pro pochopení funcionality celé aplikace je ideální provést základní rozbor hlavních kódů a vysvětlit si operace prováděné uvnitř jejich metod. [26]

7.2 Funkcionalita v kódu

Klient - ClientVehicle.js

Tato část kódu využívá několik volně šiřitelných node.js modulů:

- var temp = require('ds18b20');
- var serialgps = require('serialgps');
- var net = require('net');
- var mysql = require("mysql");
- var events = require('events');
- var fs = require("fs");
- var path = require("path");
- var ip = require("ip");

Modul ds18b20 komunikuje s teplotním čidlem pomocí protokolu one wire. Přečtené teplotní hodnoty jsou průběžně ukládány do automaticky vytvořeného adresáře a modul ds18b20 využívá metod k jejich načtení do proměnných. Adresář nese název automaticky generovaný z identifikačního čísla teplotního čidla a proto je potřeba použít metodu sensors() pro skenování identifikačních čísel aktivně připojených teplotních čidel a tyto zjištěné čísla použít k nalezení adresáře a přečtení uložených hodnot metodou temperature(). Dále je nutno zajistit stabilní napájení, v našem případě USB napájeným hubem pro zajištění dostatečného napětí v obvodu usb portu teploměru. V opačném případě čidlo vykazuje neplatná identifikační čísla a neukládá žádné naměřené hodnoty.

- var temp = require('ds18b20');

Modul serialgps pracuje se standartem NMEA celosvětově využívaným pro uniformovanou navigaci, pro námořní navigaci i pro komunikaci s navigačními satelity v GPS systému. (global positioning system) NMEA standart určuje několik typů kódů, které dohromady tvoří tzv. větu (sentence). GPS sensor pracující s těmito standarty vysílá každou sekundu tuto větu a pro její správné přečtení je třeba tyto kódy oddělit (parsing). Modul serialgps je navržen pro čtení těchto kódů v běžném připojení sensoru v sériovém módu.[19][20]

- var gps = new serialgps('COM9',4800);
  - vytvoření nového gps objektu
  - v rozšíření možno využít další metody serialist pro auto scan portu
- gps.on('data', function(data) {
  - v momentě obdržení dat provede příkazy v bloku metody
- if (sentence.sentence == 'RMC') {
  - rmc kod obsahuje zeměpisnou šířku a délku
  - v tomto bloku příkazu je také použit přepočet nutný pro obdržení přesných hodnot
  - práce s řetězcem pro oddělení označení délky a šířky (N,E,S,W)
  - práce s adresáři pro zápis hodnot, vytvoření nového pro každých 24 hodin a specifické vozidlo

- if (sentence.sentence == 'GGA') {
  - gga obsahuje údaj o nadmořské výšce

- if (sentence.sentence == 'VTG') {
  - vtg obsahuje údaj o rychlosti

- if (sentence.sentence == 'GLL') {
  - gll obsahuje aktuální gps časovou známku
  - práce se soubory pro zápis hodnot, vytvoření nového pro každou minutu a specifické vozidlo
  - uložení načtených hodnot na konec souboru každou sekundu
  - vytvoření řetězce pro přenos hodnot veličin a poslaní na server metodou .write

- function fileToDatabase(file){
  - this.con = mysql.createConnection({ host: "localhost", user: "root", password: "vertrigo", database: "world" }); connection when saving data locally
  - this.con = mysql.createConnection({ host: "192.168.0.105", user: "clientVehicle", password: "YETiH1lwR9oVQNMo", database: "world" }); vzdáleně na server přes síť, nutno vytvořit účet a použít přidělené heslo
  - this.con.query('LOAD DATA LOCAL INFILE ? INTO TABLE ?? CHARACTER SET utf8 ' + 'FIELDS TERMINATED BY ? LINES TERMINATED BY ? ', [file, 'recordings', ',', '\r\n'] mysql příkaz pro jednorázove načtení celého minutového souboru do databáze

Hlavní výhody takto navrženého přenosu:

- ✓ Dosažená konfigurace splňuje požadavky pro čtení, zápis a přenos číselných hodnot bez nežádoucích prodlev.

- ✓ Zaručuje také úspornější přenos dat s pomocí využití protokolu s několikrát menším záhlavím. Nedochází tak k opakovanému přenosu nepotřebných bajtů většího záhlaví.

- ✓ Není potřeba provozovat long pooling techniky pro udržení otevřeného spojení mezi klientem a serverem. Modul net využívá oboustranné komunikace.

- ✓ Data jsou ukládána několika způsoby najednou. V reálném čase jsou data vysílána na server aplikace, zároveň jsou ukládána na paměťové médium mikrokontroleru a také vysílána na další databázový server.

Server - App.js

Spolu s kódem index.js tvoří app.js hlavní řídící páteř a rozcestí aplikace využívající node.js middleware. Pomocí tzv. exports se stávaji funkce obsažené v těchto kódech globálně dostupné a lze se na ně odvolávat z kódů jiných.

Tato část kódu využívá několik volně šiřitelných node.js modulů:

- var express = require('express')
- var routes = require('./routes')
- var http = require('http')
- var path = require('path');
- var app = express();
- var bodyParser = require('body-parser');
- var cookieParser = require('cookie-parser');
- var session = require('express-session');
- var bcrypt = require('bcrypt-nodejs');
- var ejs = require('ejs');
- var passport = require('passport');
- var LocalStrategy = require('passport-local').Strategy;
- var net = require('net');
- var events = require('events');
- var io = require('socket.io').listen(ioport);
- var ip = require("ip");
- var Model = require('./model');

V této části aplikace probíhá inicializace serveru tcp a http. Ověření jména a hesla uživatele. Přesměrovávání na webové strany aplikace podle aktuálních žádostí klienta. Příjem a čtení datového řetězce v pravidlených intervalech. Inicializace ikony aplikace. Přeposílání aktuálně načtených dat z řetězce vyslaném mobilním klientem na vozidle k dalším připojeným klientům sledující tyto hodnoty v grafech, číselnících a mapových podkladech. Dále zde jednorázově proběhne potřebná inicializace prvků node.js frameworku Express. Dohromady využívá tato část kódu pro komunikaci server klient třech rozdílných otevřených portů, kde tcp server použije port č.8080, a http server komunikuje paralerne přes dva otevřené porty č.3000 a č.3002. Čísla portů je možné měnit, je však nutné vždy tyto porty otevřít na modemu spojující WAN s LAN typem sítě při využití veřejné internetové adresy.

- passport.use(new LocalStrategy(function(username, password, done) {
  new Model.User({username: username}).fetch().then(function(data) {
  - ověření uživatele, hesla a serializace a deserializace pro bezpečné uložení do databáze

- var favicon = require('serve-favicon');
  app.use(favicon("public/icon_offroadvehicle.jpg"));
  - inicializace ikony aplikace, framework vyžaduje uložení ilustrace ikony pouze do public adresáře

- var UserManager = require('./userManager').UserManager;
  var userManagerService = new UserManager(app);
  - inicializace middleware technologie frameworku Express

- app.get('/signIn', routes.signIn);
  app.post('/signIn', routes.signInPost);
  app.get('/signUp', routes.signUp);
  app.post('/signUp', routes.signUpPost);
  - express middleware routing navigace na metody konkrétních webových stran

- io.sockets.on('connection', function (socket) {
  console.log("IO server listens at port: "+ioport,"new visitor connected(io socket)!");});
  - otevření oboustranné komunikace s klientem pomocí frameworku socket.io

- var server = net.createServer(function(connection) {
  - otevření oboustranné komunikace pomocí modulu net a tcp protokolu

- server.listen(tcpPort, ip.address(), function () {
  console.log('Tcp server at port:'+tcpPort);});
  - spuštění tcp serveru s dynamickým zjištěním IP adresy

- http.createServer(app).listen(app.get('port'), function(){
  console.log('Express server listening on port : '+ ip.address()+' :' + app.get('port'));});
  - spuštění http serveru s dynamickým zjištěním IP adresy

Hlavní výhody takto navrženého kódu:

- ✓ využití frameworku pro oddělení vrstev aplikace
- ✓ využití frameworku pro bezpečné ověření uživatele a serializaci při ukládání do databáze
- ✓ dynamické zjištění IP adresy serveru
- ✓ nezávislý provoz pomocí využití více otevřených portů
- ✓ rychlá a jednoduchá dynamická komunikace pomocí využití frameworku socket.io

Server - Index.js
Tato část kódu využívá několik volně šiřitelných node.js modulů:

- var passport = require('passport');
- var bcrypt = require('bcrypt-nodejs');
- var Model = require('./../model');

Spolu s kódem app.js tvoří app.js hlavní řídící páteř a rozcestí aplikace využívající node.js middleware, což si můžeme představit jako oboustrannou mezistanici pro datový provoz. V obou směrech tedy při request i response je možno provést ještě další námi požadované operace a teprve potom předat data dále. Příkladem je validace uživatele při žádosti o jakoukoliv webovou stranu aplikace. V případě, že uživatel není přihlášen je automaticky přesměrován na stranu přihlášení.
V této části aplikace probíha kontrola a přeposlání konkrétních požadavků uživatele na destinaci aplikace a přidružených proměnných:

- exports.index = function(req, res){
  if(!req.isAuthenticated()) {
  res.redirect('/signIn');
  } else {
  var user = req.user;
  if(user !== undefined) {
  user = user.toJSON();
  }res.render('index', {title: 'Anavi Home', user: user}); }};
  - typový příklad provozu při žádosti uživatele o webovou stranu (zde konkrétně o hlavní stranu aplikace Home), není-li přihlášen, je přesměrován, je-li přihlášen je request přeposlán dále spolu s hodnotami proměnných title a user.

- exports.signIn = function(req, res, next) {
  - žádost o stranu přihlášení uživatele

- exports.signInPost = function(req, res, next) {
  - žádost o stranu přihlášení uživatele metoda Post

- exports.signUp = function(req, res, next) {
  - žádost o stranu registrace uživatele

- exports.signUpPost = function(req, res, next) {
  - žádost o stranu registrace uživatele metoda Post
- exports.signOut = function(req, res, next) {
  - žádost o stranu odhlášení
- exports.notFound404 = function(req, res) {
  - informační strana při nenalezení požadované destinace

- exports.event = function(req, res){
  - žádost o stranu s informacemi o zaznamenávací události s aktuálními daty

- exports.eventReplay = function(req, res, next){
  - žádost o stranu s informacemi o zaznamenávací události s daty z databáze

- exports.vehicle = function(req, res){
  - žádost o stranu s informacemi o vozidle

- exports.team = function(req, res){
  - žádost o stranu s informacemi o týmu

- exports.statistics = function(req, res){
  - žádost o stranu se statistikami záznamu měření

- exports.settings = function(req, res){
  - žádost o stranu nastavení aplikace

- exports.about = function(req, res){
  - žádost o stranu s informacemi o aplikaci

- exports.userAccount = function(req, res){
  - žádost o stranu účtu uživatele

Hlavní výhody takto navrženého kódu:

- ✓ výhody načítání kódu až v momentě kdy je potřeba,
- ✓ v momentě inicializace je celá funkce jen definovaná proměnná.
- ✓ oddělení vrstev.

Server – x.manager.js

Tato část kódu využívá volně šiřitelného node.js modulu:

- var mysql = require('mysql');

Všechny kódy x.manager.js navazují na kód index.js a přeposílají si s tímto kódem data v obou směrech, ale každý kód už obhospodařuje data pro konkrétní webovou stranu, tedy EventManager.js už řeší pouze data pro webovou stranu Event. Uvnitř těla funkcí potom volají konkrétní metody definované v následujícím kódu x.provider-mysql.js a přeposílají jim potřebné proměnné. Takto dochází k oddělení vrstev architekturou MVC. V případě, že na konkrétní webové straně nedochází ke komunikaci s databází, nepotřebuje kódy x.manager a x.provider-mysql.

V této části aplikace probíhá přeposlání požadavků uživatele na konkrétní destinaci aplikace a přidružených proměnných a předání dat zpět s odpovědí s daty z databáze:

- EventManager = function(app) {
  var EventProvider = require('./eventsProvider-mysql').EventProvider;
  var eventProvider = new EventProvider();
  - inicializace objektu aplikace

- app.set('eventProvider', eventProvider);
  - vlastní metoda node.js set, směrování na kód eventProvider
- app.get('/events', function(req, res) {
  eventProvider.fetchAllEvents(function(error, events) {
  res.send(events); - get, přeposlání pole events

- app.post('/event/events', function(req, res) {
  eventProvider.insertEvent(req.body, function(error, event) {
  res.send(event); - post, přeposlání pole events

- app.get('/event/events/:id', function(req, res) {
  eventProvider.fetchEventById(req.params.id, function(error, event) {
  res.send(event); - get, přeposlání pole events

- app.get('/eventReplay/:id', function(req, res) {
  eventProvider.fetchEventsRecordings(req.params.id,
  function(error, recordings) {
  res.send(recordings); - get, přeposlání pole recording
- app.post('/eventReplay/', function(req, res) {
  res.render('eventReplay', { title:'Anavi Event Replay', recordings:req.body});
  - přeposlání title, recordings
- app.delete('/event/events/:id', function(req, res) {
  eventProvider.deleteEvent(req.params.id, function(error, event) {
  res.send('');
  - mazání konkrétní položky, přeposlání parametru id
- exports.EventManager = EventManager;
  - inicializace kódu eventManager do middleware

Server – x.provider-mysql.js

Tato část kódu využívá volně šiřitelného node.js modulu:

- var mysql = require('mysql');

Každý kód x.provider-mysql.js navazuje na korespondující kód x.manager.js a přeposílá si s tímto kódem data v obou směrech, oba obhospodařují provoz pro konkrétní webovou stranu. Těla funkcí kódu x.provider-mysql.js potom obsahují konkrétně definované SQL dotazy směřované do MySql databáze Anavi na externím serveru obsahující uložená data aplikace.

- EventProvider = function() { - připojení na databázi na lokálním serveru při vývoji
  this.con = mysql.createConnection({host: "localhost", - definice
  user: "root",
  password: "pass",
  database: "anavi"});
  this.con.connect(function(err){ - připojení s výše uvedenou definicí proměnné con
  this.con.query('SELECT * FROM event',function(err,results){ - výpis všech záznamů
- this.fetchAllEvents = function(cb) {
  this.con.query('SELECT * FROM event',function(err,results){
  cb(null, results); - načtení do pole results a přeposlání zpět
  this.con.end(function(err) {console.log("connection closed");});- ukončení spojení

- this.fetchEventById = function(id, cb) { - vyhledání konkrétního záznamu pod daným id
  this.con.query('SELECT * FROM event WHERE id = ?',[id],function(err,result){

- this.fetchEventsRecordings = function(id, cb) { - vyhledání v zadaném časovém rozmezí
  this.con.query('SELECT r.* FROM recordings r WHERE r.timestamp > (
  SELECT e.startStamp FROM event e WHERE e.id = ?)'+'and r.timestamp < (
  SELECT e.endStamp FROM event e WHERE e.id =?)',
  [id,id],function(err,results){
- this.insertEvent = function(event, cb){ - přidání nového záznamu za nejvyšší nalezené id
  this.con.query('SELECT id FROM event WHERE id = (
  SELECT MAX(id) FROM event);',
  function(err,rows,fields){
  conn.query('INSERT INTO event
  SET id = ? , start = ? , end = ? , type = ? , startStamp = ? , endStamp = ?
  [event.id,event.start,event.end,event.type,event.startStamp,event.endStamp]
- this.deleteEvent = function(id, cb) { -smazání záznamu podle zadaného id
  this.con.query('DELETE FROM event WHERE id = ?', [id]

- exports.EventProvider = EventProvider; - inicializace kódu eventProvider do middleware

Klient – Jades.jade

Skriptovací jazyk jade je zajisté praktické moderní nářadí pro tvorbu dynamických webových stran pomocí node.js. Sám jsem si vyzkoušel jak se s takovýmto nářadí pracuje.

Má své výhody a nevýhody:

- ✓ Méně intuitivní syntax, zato ušetří opakování a zestruční psaní kódu
- ✓ Je možno použít kontrolní překladač html2jade
- ✓ Velkou výhodou a důvodem proč jsem se rozhodl se jej naučit je existence šablon
- ✓ Je ideální pro tvorbu dynamických stran

V praxi, tedy i v projektu Anavi máme možnost napsat záhlaví a šablonu pro menu jen jednou a použít je na všech dalších stranách. Dynamicky definované proměnné nesoucí jména a hodnoty konkrétních stran se potom načítají až v momentě zvolení konkrétní strany.

Také veškeré společné, potřebné podpůrné javaskripty provádějící logiku v pozadí webových stran je možno načíst jen jednou v momentě inicializace aplikace.

- layout.jade - je tedy základní šablonou obsahující načtení podpůrných javascripts
  - script(src='https://ajax.googleapis.com/ajax/libs/jquery/1.12.0/jquery.min.js')
  - link(href='https://maxcdn.bootstrapcdn.com/bootswatch/3.3.6/slate/bootstrap.min.css', rel='stylesheet')
  - script(src='http://maxcdn.bootstrapcdn.com/bootstrap/3.3.6/js/bootstrap.min.js')
  - script(src='http://ajax.googleapis.com/ajax/libs/angularjs/1.3.3/angular.min.js')
  - script(src='http://cdnjs.cloudflare.com/ajax/libs/angular.js/1.3.3/angular-route.min.js')
  - link(rel='stylesheet', href='//cdnjs.cloudflare.com/ajax/libs/bootstrap-datepicker/1.3.0/css/datepicker3.min.css')
  - script(src='//cdnjs.cloudflare.com/ajax/libs/bootstrap-datepicker/1.3.0/js/bootstrap-datepicker.min.js')
  - script(src='https://www.google.com/jsapi')
  - script(src='http://maps.google.com/maps/api/js?sensor=true')
  - script(src='https://cdn.socket.io/socket.io-1.4.5.js')
  - script(src='/fusioncharts/fusioncharts.js')
  - script(src='/fusioncharts/fusioncharts.charts.js')
  - script(src='/fusioncharts/fusioncharts.theme.fint.js')
  - script(src='/fusioncharts/fusioncharts.widgets.js')
  - script(src='/javascripts/ip.js')
  - a také obsahující šablonu pro společné menu aplikace

```
#navbar-inverse.collapse.navbar-collapse
ul.nav.navbar-nav
li(class=title=='Anavi Home'?'active':undefined): a( href="/") Home
li(class=title=='Anavi Event Info'?'active':undefined): a( href="/event") Event
li(class=title=='Anavi Vehicle Info'?'active':undefined): a( href="/vehicle") Vehicle
li(class=title=='Anavi Team Info'?'active':undefined): a( href="/team") Team
li(class=title=='Anavi Statistics Info'?'active':undefined): a( href="/statistics")Statistics
li(class=title=='Anavi Settings Info'?'active':undefined): a( href="/settings") Settings
li(class=title=='Anavi About Info'?'active':undefined): a( href="/about") About
```

Další dynamické webové strany aplikace:

- index.jade - je úvodní stranou aplikace. Tato strana i všechny ostatní jednoduchým příkazem extend layout načtou šablonu layout.jade a připojí konkrétní definici vzhledu dané strany. Úvodní strana obsahuje automatickou fotoprezentaci ilustrací z průběhu měření. Zobrazí se až v momentě úspěšného přihlášení uživatele. V opačném případě se zobrazí pouze strana přihlášení bez možnosti vstupu do aplikace.

- event.jade - je stranou pro administrativu a sledování aktuálního měření a přehrání dat z předešlých měření

- vehicle.jade - je stranou pro sledování virtuálních číselníků vozidla v reálném čase

- team.jade - je stranou pro administrativu osob s různými úrovněmi přístupů, tato strana se zobrazí v rozlišné formě administrátorovi a běžnému uživateli.

- statistics.jade - je stranou pro sledování průběhu hodnot měření v reálném v závislosti na dalších veličinách v grafových podkladech fusioncharts.

- about.jade - je informativní statická strana nesoucí základní informace o aplikaci Anavi

Tyto strany aplikace komunikují s podpůrnými skripty typu javascript, které se starají o logiku a provoz dat v pozadí aplikace. Kontrola zadaných údajů uživatelem, přeposlání a načtení dat s databází, zobrazování aktuálních hodnot měření. S využitím frameworku Bootstrap a Bootswatch je vzhled všech stran aplikace responsivního formátu, tedy adaptuje se automaticky na velikost obrazovky ve které je aktuální strana otevřena. Také menu je adaptabilní a mění se podle typu přístroje.[24][25]

Klient – Javascripts.js

Popis funkčnosti nejvýznamnějších kódů typu javascript provádějících tzv. backend logické operace v pozadí konkrétních webových stran nacházejících se na klientovi obsahuje následující výčet:

- userAccountManager-client.js – tento kód obsahuje funkci onUpdate(id), která zajišťuje logiku pro aktualizaci účtu uživatele samotným uživatelem. Při prvotní registraci se vytvoří účet uživatele, který je dále takto možno doplnit či pozměnit.

- userManager-client.js – tento kód obsahuje logické operace pro vykonání základních databázových dotazů CRUD. Komunikuje tak nepřímo s kódem userProvider-mysql.js kde dochází k přenosu jednoho záznamu databáze, tedy uložení nového, nebo úprava existujícího.

- eventsManager-client.js - tento kód obsahuje podpůrné logické operace pro komunikaci s databází, podporu pro změnu dynamického vzhledu strany v závislosti na módu zobrazování dat, funci pro zajištění opakovaného přehrání záznamu měření.

- eventsDateValidation.js – tento kód využívá malého volně šiřitelného kódu tzv. gadget, který definuje vzhled a základní funkce kalendáře. Tedy není nutno programovat svůj vlastní kalendář ale zakomponovat tento gadget na webovou stranu. Logika našeho kódu se potom stará o načtení hodnot zadaných přes kalendář gadget a jejich verifikaci.

- eventMap.js – tento kód také využívá volně šiřitelné předdefinované části kódu od společnosti google pro podporu mapových podkladů. V tomto případě náš kód zodpovídá za správné nastavení vzhledu mapových podkladů, správné doručení zobrazovaných hodnot a interval zobrazení.

- eventMapReplay.js – tento kód společně s předešlým kódem eventMap.js obsahuje také podpůrnou logiku matematických operací s hodnotami proměnných zobrazovanými v mapových podkladech. Tyto matematické operace je možno umístit na kódy prováděné serverem, nebo klientem. Zvolil jsem klienta z důvodu menšího zatížení kapacity serveru.

- vehicleGraphs.js - tento kód komunikuje se serverem pomocí dalšího otevřeného portu v našem případě 3002, přes který získává aktuální data v reálném čase přeposlaná serverem aplikace přímo z klienta umístěného na vozidle v terénu. Logika kódu zajistí okamžité zobrazení hodnot v momentě jejich přijetí. Dále využívá předvytvořených kódů pro zobrazování grafů knihovny fusionchart.

7.3 Konfigurace

Základní prioritou funkčnosti aplikace je důraz na rychlost přenosu dat a minimální zpoždění zobrazení. Konfigurační řešení z těchto požadavků vychází a obsahuje tyto prvky:

- ✓ použití úsporných typů přenosu TCP a WS sockets
- ✓ menší overhead v záhlaví bez potřeby long pooling
- ✓ přizpůsobení intervalu ukládání dat do databáze
- ✓ rozložení výpočetních operací také na stranu klienta
- ✓ ušetření výpočetního výkonu serveru
- ✓ komunikace klient – server přes více otevřených portů

7.4 Správa dat

Více úrovní přístupu zajišťuje možnost rozdělení privilegií mezi uživatele. Po založení běžného uživatelského účtu má možnost uživatel upravit svůj účet a prohlížet záznamy o měření, vozidlech a osobách týmu. Administrátor má možnost spravovat všechny uživatelské účty, editovat a mazat záznamy o měření, vozidlech a osobách týmu. S využitím frameworku Passport.js není bez platného uživatelského účtu možný přístup k datům aplikace.

Naměřené hodnoty veličin jsou ukládány s využitím modulů fs a path do souborů na fyzické paměťové médium v mikrokontroleru ve vozidle. Paralerně probíhá ukládání dat do databáze mysql pomocí modulů mysql. Databáze je typu myISAM a ukládání probíhá ve smyslu vložení celého souboru jednorázově.

Rychlost přenosu aktuálních dat má nejvyšší prioritu, kde odděleným transportním řetězcem data putují z klienta data sbírající přes server bezprostředně do prohlížeče klienta data zobrazující.

7.5 Vstupy a výstupy

V oblasti vstupů aplikace pracuje s hodnotami základních veličin dostupných při pohybu vozidla v terénu. Aplikace aktuálně využívá sensorů od firmy Microsoft a Dallas Semiconductor a pracuje s hodnotami veličin reálného světa, tedy není potřeba simulace a vytváření virtuálních dat. Moduly serialgps a ds18b20 zabezpečí načtení dat do proměnných vstupního kódu nacházejícího se na klientu ve vozidle.

V oblasti výstupů aplikace pracuje s několika typy dat. V prvé řadě zobrazuje reálné hodnoty jednotlivých veličin získané přímo ze sensorů na vozidle. Dalším zdrojem dat pro zobrazení na výstupu aplikace jsou data z databáze, která se zobrazují v přehledných tabulkách. Nesou informace o vozidlech a osobách převážně pro administrační účely. Dále samotné naměřené hodnoty uložené do databáze je možno znovu zobrazit a opakovaně si projít průběh měření v mapových a grafových podkladech.

8. TESTOVÁNÍ

V momentě, kdy se při vývoji systému podařilo dosáhnout bodu kdy samotný projekt naplní požadavky zadání a pracuje bez problémů v testovacím prostředí, je třeba přejít k testování v reálném skutečném prostředí pro které je systém navrhován. Zakladní, prvotní testovaní při vývoji probíhalo na zařízení laptop Lenovo AMD A4-5000 a to v lokální LAN síti, kdy všechny scripta byla lokalizována na zmíněném zařízení. Další stupeň testování bylo umístění sensorů na samostatný micropočítač Raspberry Pi2, na kterém již v reálném nasazení zůstanou. Raspberry takto komunikovalo přes LAN síť a to pomocí usb adapteru TP-Link TL WN722N.

Dalším krokem byla instalace utility pro mobilní připojení a výměna usb adapteru pracujícím na lokální síti za usb adapter Huawei E3131 a spuštění na WAN veřejné síti. Také zajištění veřejné adresy a povolení provozu přes otevřené porty, které aplikace využívá, tedy port forwarding. Hlavní přístupový bod aplikace na domácí stranu je povolen přes port 3000, port 8080 zabezpečuje komunikaci TCP protokolu nesoucí řetězec dat sensorů na vozidle a port 3002 přepošle zobrazovaná data ke klientům do prohlížečů. Taková konfigurace je již prakticky identická s konfigurací systému pro reálný provoz. [15]

Při testování aplikace v terénu, tedy při skutečném pohybu vozidla se projevil očekávaný problém ztráty připojení k mobilní síti v místech s horším pokrytím. Tento problém byl vyřešen instalací modulu forever.js, který spouští hlavní kód aplikace i kód klienta produkující data v pozadí a zajistí automatický restart. Bylo třeba také zajistit dostatečné napětí pro micropočítač Raspberry Pi2 a to napětovým zdrojem 2A a více a USB hub s externím napájením. V opačném případě docházelo k různým anomáliím, například špatné označení / pojmenování vytvářených adresářů pro ukládaná data teplotních sensorů a kompletní restart jednotky Rpi2. Pro reálný provoz byla použita čidla pro navigaci od firmy Microsoft, produkující data jako pozici, nadmořskou výšku, rychlost, časovou známku a dva druhy teplotních čidel DS18B20. Mobilní verze a responzivní vlastnosti byly otestovány na zařízeních LG400 a Apple iPad mini 2 MF080LL/A.

Při provozu na veřejné síti bylo ještě potřeba zajistit Google API key, tedy unikátní klíč od firmy Google pro přístup k mapovým podkladům. Tímto se firma Google chrání proti neoprávněnému přístupu k mapovým podkladům v komerčních verzích projektů.

9. ZHODNOCENÍ VÝSLEDKŮ A MOŽNÁ ROZŠÍŘENÍ

Protože vývoj aplikace probíhal v malých krocích, výsledné testování přineslo očekávané výsledky bez větších dramatických překvapení. V průběhu vývoje i testování se potom objevily další potřeby a možnosti rozšíření. Přestože je aplikace navržena pro detailní zobrazení dat a rozvržena do několika základních bloků resp. webových stran podle druhu dat, ukázalo se, že praktické využití by našla další webová strana zobrazující souhrn více druhů dat.

Momentálně v první verzi aplikace je struktura kódu zajišťující datový tok uzpůsobena tak, že je velice snadné rošíření o další sensory. V tomto místě se zrodila myšlenka vytvořit pomocnou knihovnu sensorů, ze které by si uživatel jen vybral další sensor, pojmenoval si jej a aplikace by automaticky tento sensor našla a začla z něj odebírat data. Protože dnes je každý sensor unikátní integrovaný obvod a většinou až komplikovaný mikroprocesor, bylo by třeba zmapovat standarty komunikačních protokolů rozčlenit je do skupin podle stejných druhů a uložit do knihovny.

Technologie Google Maps Api umožnuje zobrazovat pole pozic vozidel, tedy v případě potřeby rozšíření je možná úprava kódu a zobrazení více vozidel ve stejném čase, protože každé testované vozidlo má své identifikační číslo.

Přestože aplikace zpracovává jak data v reálném čase a vytváří odhady o dalším vývoji hodnot veličin, dovede zobrazit a podrobně analyzovat data uložená v databázi, zrodila se myšlenka na další praktickou funkci. Při spojení dat uložených v databázi a dat aktuálně čtených, či dokonce pro další krok odhadovaných, by se dala analyzovat situace v daném konkrétní bodě. Tedy například při závodění na okruhu, by tato funkce byla schopna porovnat aktuální a uloženou hodnotu veličiny na stejném místě.

10. ZÁVĚR

Návštěvou místního automobilového výrobce Tatra trucks a zhlédnutím testovacího okruhu, laboratoří a způsobu práce se získanými daty se zrodila inspirace k průzkumu téma rozboru a vizualizace dat z čidel osobních a nákladních automobilů. Teprve až po pečlivém prostudování tohoto poměrně širokého téma vznikla konkrétní myšlenka, jasnější specifikace zadání, ideální skupina koncových uživatelů a samotný cíl bakalářské práce. Souhrn studia tématu je obsažen v kapitolách jedna až tři. Je zde popsán počátek monitorování dat získávaných z čidel automobilů a původní účel. První oficiální standarty kodů včetně komunikace po sběrnici, rychlosti a kapacity přenosů. Dále bakalářská práce kompletně představuje systém elektronických čidel a ECM konkrétní pohonné jednotky nákladních automobilů s jejich fyzickým umístěním a grafickým popisem. Následuje průřez změn a obecná aktuální situace v oblasti využití elektronických modulů v automobilovém průmyslu dnes. Další kapitoly věnují pozornost přenosu získaných dat mimo vozidlo a alternativním řešením ke standartní sériové produkci.

Dokončením této části studia se formuje konečná myšlenka a konkrétní specifikace zadání, jemuž se následně věnuje kapitola čtyři. Tvorbu aplikace předcházel výběr nejlépe se hodících programovacích nářadí, způsob jejich použití a způsob konečného propojení jednotlivých komponent. Kapitoly věnující se analýze, návrhu a implementaci popisují detailně tuto část přípravy a tvorby. V průběhu dokonce došlo k rozhodnutí zaměnit programovací technologie z důvodu lépe se hodící varianty. Konečná aplikace tedy používá z mého pohledu lépe se hodící programovací technologie a dosáhla naplnění požadavků specifikovaných v zadání. Je schopna bez nutnosti registrace a paušálních poplatků přenést, analyzovat a zobrazit v reálném čase data čtená z čidel instalovaných na vozidle uživatele. Podařilo se navrhnout způsob ukládání dat jež minimálním možným způsobem ovlivňuje chod celé aplikace.

Softwarové řešení vychází z konkrétních požadavků koncového uživatele na výběr fyzických komponent, kde hodnota použitých elektronických součástek je řádově až desetkrát menší oproti těm sériově dodávaným, jejich dostupnost několika násobně lepší a konečně jednoduchá komunikace s nimi a srozumitelné řešení jejich zapojení. Další studentské projekty nejen z fakulty Elektrotechniky a informatiky tak mohou použít tuto aplikaci jako výchozí stavební část pro projekty týkající se monitorování veličin pohybujících se automobilů. Také autoři středních a menších nestandartních projektů mohou experimentovat s touto aplikací a hardwarovou částí na které fyzicky poběží právě z důvodu nižší ceny komponent.

Aplikace je dále rozšiřitelná v oblasti vlastních funkcí se získanými daty prováděných podle nových potřeb uživatelů i okrajových funkcí pro komunikaci s vnějšími komponenty systému. Zde je možné systém rozšiřovat začleňováním nových specifických node modulů potřebných pro konkrétní funkčnost, či tvorbou vlastních modulů a jejich integrací.

11. REFERENCE

[1] Measurement using a thin film sensor, *Yuji Mihara, Tokyo city university*[online]. 2016 [cit. 2016-05-15] Dostupné z: http://www.miics.net/archive/getfile.php?file=219

[2] Future advances in body electronics, *Freescale, Bodyelectrwp*[online]. 2016 [cit. 2016-05-15] Dostupné z: http://cache.nxp.com/files/automotive/doc/white_paper/BODYDELECTRWP.pdf

[3] DDTD S60, *Lucian E.Marin Sam, Detroitdieselenginesinfo*[online]. 2016[cit. 2016-05-15] Dostupné z: http://www.detroitdieselengines.info

[4] Detroit Diesel Diagnostic Link 7.04, *AutoCD*[online]. 2016 [cit. 2016-05-15] Dostupné z: http://www.autocd.ru/DDDL7readme.pdf

[5] Detroit Diesel Diagrams, *Detroit Diesel, DDCSN*[online]. 2016 [cit. 2016-05-15] Dostupné z: https://ddcsn-ddc.freightliner.com

[6] Complex Car Software Becomes the Weak Spot Under the Hood, *David Gelles, Hiroko Tabuchi, Matthew Dolan, NYTimes*[online].2016 [cit. 2016-05-15] Dostupné z: http://www.nytimes.com/2015/09/27/business

[7] Real-Time vehicle performance monitoring with data integrity, *Will Jenkins, Mississippi State University*[online]. 2016 [cit. 2016-05-15] Dostupné z: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.461.1689&rep=rep1&type=pdf

[8] CarSim mechanical simulation, *Thomas Lenhart, Mechanical Simulation*[online]. 2016 [cit. 2016-05-15] Dostupné z: https://www.carsim.com/products/carsim

[9] The multibody dynamics simulation solution, *Jeffrey Graff, MCS Software*[online]. 2016 [cit. 2016-05-15] Dostupné z: http://www.mscsoftware.com/product/adams

[10] Tesis Dynaware solutions, *Dr.Marita Irmsher, Tesis Dynaware*[online]. 2016 [cit. 2016-05-15] Dostupné z: http://www.tesis-dynaware.com/en.html

[11] Tatra areál Kopřivnice, *Martin Bednarz, Tatra trucks a.s.*[online]. 2016 [cit. 2016-05-15] Dostupné z: http://www.tatra.cz/

[12] Automotive Sensors, *John Turner Ph.D, Momentum Press*, 2009, ISBN-10: 1606500090, 2016 [cit. 2016-05-15].

[13] SAE International Journal of Passenger Cars - Electronic and Electrical Systems, *SAE International, SAE*, 2016, ISSN 1946-4614, 2016 [cit. 2016-05-15].

[14] Wireless Networks, *Clint Smith, Daniel Collins, McGraw-Hill Education,* 2014, ISBN-10: 0071819835, 2016 [cit. 2016-05-15].

[15] The Official Raspberry Pi Projects Book, *Russell Barnes/team, Pi Magazine*, 2015, ISSN: 2051-9982, 2016 [cit. 2016-05-15]

[16] OBD-II PIDs, *Wikipedia, the free encyclopedia*[online]. 2016 [cit. 2016-05-15] Dostupné z: https://en.wikipedia.org/wiki/OBD-II_PIDs

Reference – pokračování

[17] A controller area network, *Wikipedia, the free encyclopedia*[online]. 2016 [cit. 2016-05-15]. Dostupné z: https://en.wikipedia.org/wiki/CAN_bus

[18] OBDuino, *Wikipedia, the free encyclopedia*[online]. 2016 [cit. 2016-05-15]. Dostupné z: https://en.wikipedia.org/wiki/OBDuino

[19] NMEA 0183 Standard, *NMEA, National Marine Electronics Association* [online]. 2016[cit. 2016-05-15] Dostupné z: http://www.nmea.org/content/nmea_standards/nmea_0183_v_410.asp

[20] NMEA data, *Joe Mehaffey, Jack Yeazel, Sam Penrod, Allory Deiss, GPSinformation*[online]. 2016 [cit. 2016-05-15] Dostupné z: http://www.gpsinformation.org/dale/nmea.htm

[21] Node.js in Action, *Mike Cantelon, Manning Publication,* 2013, ISBN-10: 1617290572, 2016 [cit. 2016-05-15].

[22] Node.js the Right Way, *Jim R. Wilson, Pragmatic Bookshelf,* 2013, ASIN: B017WQOUE6, 2016 [cit. 2016-05-15].

[23] Web development with Node.js and Express, *Ethan Brown, O'Reilly Media,* 2014, ISBN-10: 1491949309, 2016 [cit. 2016-05-15].

[24] A Software Engineer Learns HTML5, JavaScript and JQuery, *Dane Cameron, CreateSpace Independent Publishing,* 2013, ISBN-10: 1493692615, 2016 [cit. 2016-05-15].

[25] JavaScript: The Ultimate Beginner's Guide!, *Andrew Johansen, CreateSpace Independent Publishing,* 2016, ISBN-10: 152373082X, 2016 [cit. 2016-05-15].

[26] NPM modules, *NPM, NPM library*[online]. 2016 [cit. 2016-05-15] Dostupné z: https://www.npmjs.com/

[27] DMXzone bootstrap manual, *DMXzone, DMXzone.com*[online]. 2016 [cit. 2016-05-15] Dostupné z: https://DMXzone.com

[28] Learning MySQL, *Seyed Tahaghogbi, Hugh E. Williams, O'Reilly Media,* 2006, ISBN: 978059600864-2, 2016 [cit. 2016-05-15].